COMP07027 Introduction to Programming

Practice Programming Project – 2013/14

Project Work

This project is an individual programming project for you to try for practice over the next few weeks when we have no classes. You should start this in the lab in week 12 but the idea is give you some programming to do between now and the beginning of trimester 2 so you do not forget the things that you have learned in trimester 1, and to give you a chance to apply what you have learned to a larger and more interesting problem than the exercises in the lab you have been doing.

You should submit your work using the **Practice Programming Project** link in the Assignments area of the module site on Moodle no later than **Monday 20th January 2014 at 4:00 pm**. See the submission notes at the end for more details of what you should submit. You will not receive a mark for this project but you will get some feedback.

This project description is taken from the 1999 text by Duane A Bailey and Duane W Bailey, Java Elements: Principles of Programming in Java. (McGraw-Hill Education)

Baby, Crab and Cone Puzzle

You are to develop a program to validate potential solutions to a classic puzzle[1] recast as Mama's puzzle of the baby, crab, and cone. As the story goes, Mama headed off to the ice cream shop on a hot summer day with her delightful baby and pet crab. On the way she crossed a river with the aid of a ferry boat that, rowed by herself, could carry one other. A little thought demonstrates that this was no minor obstacle—it required three trips in the ferry to get everyone across (the crab it seems, was no great swimmer). By the time she got to the shop, she was clearly ready to make a purchase.

Having purchased the largest cone possible, Mama set back toward home with all in tow: baby, crab, and cone. At the river, however, she faced a dilemma: in what order should she row everyone across in the ferry? The difficulty was that since only one of the baby, crab, or cone could be ferried across (it must have been a huge cone), two must be left behind. Unfortunately, if the baby was left unattended with the crab, violence of some form would surely ensue. If the baby was left with the cone, it would overindulge. Thus, each trip of the ferry had to be made with care, making sure that the baby was never left with either of the other two on same shore.

[1] The farmer's puzzle of the chicken, fox and the corn

To facilitate Mama's journey, you are to write a planning program of sorts, that allows the harmless simulation of the various possibilities. We have in mind a program with output similar to the following:

Welcome to the Puzzle of the Baby, the Crab, and the Cone.
One hot summer day, Mama crossed the river to the left bank take a break from programming and buy an ice cream. She brought her baby and her pet crab. On the return she came to the river and was faced with a dilemma: how could they all cross without disaster? The ferry holds Mama and just one other item. The problem is that
* if the baby is left with the cone, it spoils its dinner.
* if the baby is left with the crab, the crab bites the baby.
Please help Mama make her journey across the river!
Mama is on the left.
The baby is on the left bank.
The crab is on the left bank.
The cone is on the left bank.
Who would you like Mama to cross with?
(type 0 for no one, 1 for baby, 2 for crab, or 3 for cone)
2
Mama crosses the river with her pet crab.
Mama is on the right.
The baby is on the left bank.
The crab is on the right bank.
The cone is on the left bank.
Oh no! The baby ate the ice cream.
We'll have to stop and help Mama and her family.
Undoubtedly, she'll need your help again.

As you can tell, the program keeps track of the location of each of the parties, perhaps by a boolean. After describing the state of the situation, the facilitator is asked to determine who rides in the ferry with Mama. As Mama leaves shore, the various lethal conditions are checked for and the simulation is potentially terminated. If, however, everyone eventually makes it to the destination (here, the "right bank"), the puzzle is solved. A good program would point out that the solution might have been faster if more than seven moves were required.

Procedure. This is quite a complex program, so the following steps should be considered in the design of your program. Once designed, the program is more easily coded:

1. Identify all the variables that will be needed. Variables help to keep track of the state of the program. If something must be remembered, it must be accounted for as a variable. For example, it might be useful to keep track of the number of times the ferry has crossed the river.

2. For each variable, identify its type, and if necessary, its initial value. This

should complete your declarations.

3. Break the program down into several sizeable pieces. It's not important to know immediately how each piece is implemented, but it is necessary to identify logically distinct portions of the program. For example, the instructions must be written at the beginning, the results at the end, and in the middle it is necessary to print the location of all the participants. There may be other parts, as well.

4. Identify those parts of the program that are part of a conditional or looping construct, remembering the features that distinguish each of the loop types.

5. Our informal design is completed if a reasonable person can follow the logical states of the program without doing anything that seems contrary to the imagined execution of the program.

6. Armed with a design, it is now possible to implement each of the logical components in a programming language. It is often the case that the design had to be augmented to take into account the finer details. As you implement each of the logical components of the program, you may find it useful to test to see if the program works as expected. For example, you run the program to see if the instructions and initial state is printed correctly. The intermediate steps also help to ward off any unforeseen problems as early as possible.

7. Once completed, test your program thoroughly. Does it work in short simulations (as seen in the one above)? Does it work when you follow the optimal solution? Does it work, even if Mama makes far too many trips? What happens if the number 4 is provided as input to the question?

Thought questions you might ask in reflecting on your work:

1. In your finished program, are there any unnecessary variables?
2. Are there any statements that don't get executed?
3. Return to this code a couple of days after you finish it. Are there improvements you can see after this fresh view of your code?
4. If crabs liked ice cream (they don't), could the puzzle be solved? (Assume Mama eats neither the crab nor the cone in transit.) If so, how? If not, why?

Your program should address the following points:

Point No.	Description (the facilities you need to implement)
1	The program defines a suitable set of variables to keep track of the location of Mama, Baby, Crab and Cone.
2	The program follows the above specification in providing a description of the problem, the initial state (Mama, Baby, Crab and Cone all being on the left bank of the river) and in prompting the user to choose which item, if any, Mama is to cross the river with.
3	The program gets the user input and, if it is valid, updates the program state so that Mama and the chosen item (if any) cross the river to the opposite bank. If the input is not valid, the program displays an appropriate message and the program state is not updated.
4	After each crossing, the program displays where Mama, Baby, Crab and Cone are now located.
5	After each crossing, the program checks if disaster has occurred (Baby left alone with Crab or Baby left alone with Cone) and if it has displays an appropriate message and stops the puzzle.
6	After each crossing, the program checks if the puzzle has been solved (Mama and all items safely on the right bank) and if it has displays an appropriate message and stops the puzzle.
7	As it is a hot summer's day, the ice cream will eventually melt if not eaten. Have the program count the number of crossings and set a threshold at which the ice-cream will melt if the river has not been successfully crossed. Decide how this affects the simulation.
8	Give some thought to the structure of the program. Think about using subroutines to validate the input, to check the program state, and to update the program state. Also think about what data structures you could use to simplify the logic and capture the program state. Document your program with comments in the code. At the very least, there should be comments against each variable declaration to explain what the variable is for and above each subroutine explaining what the subroutine does. Include a comment at the beginning of the program explaining what the program does.

What you should submit

To submit your project, upload it using the link in the Assignments subpage of the Moodle site for the module.

The submission for the project should include the following:

1. A ZIP file containing all of the source code files that make up your project (including any additional library files you have used or created). As a rough guide to preparing your project for submission:
    a. Make sure ALL of the project's source files are in the project's folder. If any are not in that folder, move them or get help to do so.
    b. Go up to the containing folder and create a ZIP file of your whole project. Assuming you're using Windows XP, Vista or Windows 7, just right-click on the folder and select Send To..Compressed (zipped) folder). For other operating systems, you may need to look up the correct method for creating a ZIP file – email for help if you need it.

2. Submit the ZIP file using the **Practice Programming Project** link in the Assignments area of the module site on Moodle no later than **Monday 20th January 2014 at 4:00 pm**.