Lab Assignment #1: At the dice table
Control structures, functions and stepwise refinement

Examination 1: 2021-11-19

Introduction

The purpose of this lab assignment is to practice working with control structures and functions, using stepwise refinement for program design. You will implement three simple dice games, and also incorporate the textbook code for the game Craps in your program. The finished program will also have a simple menu system that lets the user navigate between different options.

Getting started

Read up on sections 5.11 – 5.12 in the course book, regarding random numbers and simulating dice. Also, carefully study the textbook example game Craps (Figure 5.14) and associated explanations.

Description of program functionality

When your program is run, the user will have four options to choose from, shown in a menu like this one:

```
Which game would you like to play?
   Enter 1 for Craps
   Enter 2 for The Abyss
   Enter 3 for Pig
   Enter 0 to Quit
Your choice?
```

The program must perform input checking on the chosen value. If the user input anything else than 0-3, an error message should be displayed and the menu shown again. If 0 is entered, the program should terminate, whereas options 1-3 starts the chosen game. After this game is finished, the program should return to the main menu, i.e. execution only terminates when the user chooses 0 from the main menu.

Option 1 – Craps

When the user selects this option, the program should perform exactly the same functionality as the original craps program, i.e. simulate one round of craps against the computer.

Lab Assignment #1: At the dice table
Control structures, functions and stepwise refinement

Option 2 – The Abyss

This is a two-player game, where the user plays against the computer. The players take turn rolling one die and the aim is to be first to the sum 26. However, there are some obstacles on the way…

1) If a player gets the sum 13, she falls into the abyss and loses the game.
2) If player X gets to a number between 1 and 12, and the player Y is on exactly that number, player X is sent back down to 0.
3) If player X gets to a number between 14 and 24, and the player Y is on exactly that number, player X is sent back down to 12.
4) If player X gets to number 25 and the player Y is on exactly that number, player X is sent back down to 14.
5) To finish, a player has to roll exactly the number required to get 26 from her current sum.

Option 3 – Pig

In this option, the user should play the dice game of Pig (Etthundra in Swedish) against the computer. The rules of the game are given on Wikipedia[1] as follows:

Each turn, a player repeatedly rolls a die until either a 1 is rolled or the player decides to "hold":

- If the player rolls a 1, they score nothing and it becomes the next player's turn.
- If the player rolls any other number, it is added to their turn total and the player's turn continues.
- If a player chooses to "hold", their turn total is added to their score, and it becomes the next player's turn.

The first player to score 100 or more points wins.

[1] https://en.wikipedia.org/wiki/Pig_(dice_game)

Lab Assignment #1: At the dice table
Control structures, functions and stepwise refinement

Step 1

The first task is to implement the main menu. We will use a main loop, with the following structure:

```
void main(void){
    int answer;

    while (true) {
        displayMainMenu();
        printf("Your choice? ");
             // receive input
             // handle options
    }
}
```

One of the advantages of using numbers as menu options is that we can use a switch statement to select the correct option and have the default handle all invalid choices, like this:

```
default: puts("Please enter a valid option"); break;
```

The break statement last in the default clause will display the menu again is an invalid option is chosen. However, you need to figure out how to exit the program when the user inputs 0.

Now, start coding the main menu according to the description. Also, for each of the three menu options, create functions that will be called from the switch statement. They should have the following names and prototypes:

```
void craps(void);
void pigs(void);
void twoDicePigs(void);
void abyss(void);
```

For now, just put a print statement in each function body, to indicate that the function call has worked.

We now have code for the top level of the step-wise refinement. Now test your program, with regards to input from the main menu, calls to the appropriate function from the switch statement and quitting the program.

Lab Assignment #1: At the dice table
Control structures, functions and stepwise refinement

Step 2

Here, you will incorporate the existing code for Craps into the program. After carefully studying the program in Figure 5.14, think about which parts of the code that should go where in your program. The aim is to have the code run exactly as in Figure 5.14 when the craps function is called. The only modification of functionality that you are allowed to make is to introduce a sense of tension by pausing the game at one or more points in the simulation.

Test your program again, after incorporating the craps game. Menu option 1 should now run this game, whereas the other options should call empty functions as before.

Step 3

In this step you must implement the game of The Abyss, as described above. The human player always starts the game. Think carefully about the game logic and employ stepwise refinement to break down the problem into smaller parts. Take special care to determine an appropriate condition for the main loop, that determines when gameplay stops.

As this game does not contain any choices for the players, defining the logic is only a matter of capturing the game rules with control structures and keeping track of the scores. A suggestion is to first develop your program without any checks on colliding numbers, and then introduce these checks one by one, carefully testing each case, before proceeding to the next.

Step 4

In this step you must implement the game of Pig, as described above. The human player always starts the game. Think carefully about the game logic and employ stepwise refinement to break down the problem into smaller parts. Take special care to determine an appropriate condition for the main loop, that determines when gameplay stops.

For this game, you must implement a computer opponent to the human player and this opponent must have a strategy. An optimal strategy for the game has been computed[2], but you should implement the following simpler variant:

- The computer always throws the dice twice, unless its rolls a one on its first throw, of course.
- After two rolls, a randomized decision is taken whether to throw the dice again. You are free to determine this probability, either by a static value or by some formula depending on points, how many throws etc. You only have to implement the possibility of throwing a third time.

[2] See, Todd W. Neller and Clifton G.M. Presser. Optimal Play of the Dice Game Pig, The UMAP Journal 25(1) (2004), pp. 25–47

Lab Assignment #1: At the dice table
Control structures, functions and stepwise refinement

Hints on stepwise refinement:
Since the human player when playing one turn will have choices (continue or not) determined via input, and the computer will have a hard-coded strategy for playing one turn, it is highly recommended that you develop separate functions for these. These functions can then be called in a game loop, where human and computer take turns until one of them reached 100.

In addition to the code, you must also produce a short description of how you divide your program into functions and give motivation for your choices. This description should be ½-1 page long.

Requirements

- For options 2 and 3, you must write the code yourselves. Code snippets from the textbook and/or lectures can of course be used, but it is not allowed to find code on the internet.
- Furthermore, you are only allowed to use the constructs covered so far in the course, i.e. no modules, arrays, pointers or such. If you think some parts may have been easier to design and implement with these tools, you are free to comment on it in your attached document.
- The program that you submit must compile and run without modification. This means that any code that gives compilation or runtime errors must be commented out.

Grading criteria

You will receive the grade pass or fail.
In order to pass, the program must be complete, i.e. have the full functionality described above. In addition, the program must be well-designed and well-coded, especially with regards to:

- How it is broken down into functions (one function – one task), where you strive to separate i/o and logic. Also, reuse of functions whenever possible.
- Use of control structures - clear and tidy.
- Adequate data types and consistency in this between functions
- Naming of functions and variables must be in accordance with the principles presented in the course book
- Comments for each function, describing its task, input and output
- Comments for control structures, where appropriate.

Submission

You must submit your solution via Canvas, in a zip file containing source code, executable and documentation. Deadline for submission is Friday 19th of November at 17.15

Re-examination deadline is January 31st, 2022.