

Lab 3: Tower of Hanoi

In Lab 3, you are required to design and implement a game: Tower of Hanoi.

0 The puzzle

The Tower of Hanoi is a famous mathematical puzzle. It consists of three rods and a number of disks of different sizes. Initially, all disks are in the first rod in ascending order of size (i.e., the smallest is at the top). The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No larger disk may be placed on top of a smaller disk.

The classical way to solve the puzzle is recursion. The algorithm could be described as the following pseudocode:

```
function hanoi(n, A, B, C) { // move n disks from rod A to rod B, use rod C as
a buffer
    hanoi(n - 1, A, C, B);
    move(n, A, B); // move the nth disk from rod A to rod B
    hanoi(n - 1, C, B, A);
}
```

1 Lab Description

Here are some concepts that you need to know first.

- Canvas: The UI of the game.
- Rod: There are 3 rods (numbered 1, 2, 3) in the canvas. Disks are pushed and popped among them. The source rod is 1 and the target rod is 2. Rod 3 acts as a temporary rod.
- Disk: Each disk is of different sizes. In the beginning, they are all pushed into Rod 1 in ascending order of size.

You should implement the Hanoi game program with the following functionality:

1. **Initial Phase:** When the program starts, it firstly asks the player for the number of disks. The input should be an integer in [1, 5]. If input is `q`, quit the game. Other invalid input should be ignored.
2. **Normal Mode:** Then, print the canvas in the console, and the game starts. The program continuously asks for commands until the player wins. The canvas will be printed after each move. Command format: `from to`. Invalid input should be ignored.
3. **Auto Play Mode:** If the game receives command `0 0`. Auto-play mode will be activated. The game will automatically continue without player's commands. The canvas, as well as the

auto-generated command, will be printed after each move.

Requirements

- You have to implement the game using **Object-Oriented** programming. Hope the following materials will help you if you're not familiar with OOP.
 - [Object Oriented Design](#)
 - [Object Oriented Programming in C++](#)
- Containers in STL (`std::vector`, `std::stack`, `std::list`, `std::queue`) are **NOT** allowed to use. Please implement your own `Stack` or `Queue` if needed.

2 Guidance

2.1 Drawing the Canvas

We have offered you a file, `termio.h`. It might be helpful for you to deal with the canvas printing work.

- `CANVAS_WIDTH` and `CANVAS_HEIGHT`: the width and height of the canvas.
- `Clear()` helps you clear the screen. Using it before `Draw()` will improve the user experience.
- `Draw()` helps you draw the buffer to console.
- `ResetBuffer()` fills the buffer with spaces.

2.2 Auto Play Mode

During the game, players may enter auto-play mode in any state. After entering into auto-play mode, you need to do two things.

- Restore to the beginning state (simple idea: use a stack to log the player's commands and undo these commands when entering auto-play mode).
- Auto-play the game from the beginning state.

2.3 Output format

- For the sake of simplicity, the disk is represented by `*`.
 - The number of `*` that you should draw is calculated by `2 * disk_size + 1`. For example, if there are five disks in rod1, then the number of `*` is `3, 5, 7, 9, 11`. The middle `*` is vertical aligned with the rod.
 - You should draw disks from the bottom of the rod. The two examples below show the canvas when there are 5, 3 disks at the beginning.
 - The position of the rods should remain unchanged regardless of number of disks.
 - We have defined `CANVAS_WIDTH` and `CANVAS_HEIGHT` for you, you should not change the value.

```

      |
    ***
      |
    *****
      |
    *****
      |
    *****
      |
    *****
      |
    *****
    -----|-----|-----|-----

```

```

      |
      |
      |
      |
      |
    ***
      |
    *****
      |
    *****
      |
    *****
    -----|-----|-----|----- // this line should be unchanged
    regardless of number of disks

```

- Upon winning the game (either from normal mode or auto-play mode), first print "Congratulations! You win!", and then start a new round of game.

```
std::cout << "Congratulations! You win!" << std::endl;
```

```
std::cout << "How many disks do you want? (1 ~ 5)" << std::endl;
```

- To improve user experience, you don't have to clear screen under auto-play mode.
- About other output formats, please see the example below.

3 Example

3.1 Example1

3.1.1 Initial Phase

```

./hanoi
How many disks do you want? (1 ~ 5)
2
      |
      |
      |
      |
      |

```

```

      |
      |
      |
    ***
      |
    *****
-----|-----|-----|-----
Move a disk. Format: x y

```

After telling the program about the number of disks, we come to the normal mode.

3.1.2 Normal Mode

```

1 3
      |
      |
      |
      |
      |
      |
      |
      |
      |
      |
    *****
-----|-----|-----|-----
Move a disk. Format: x y
1 3
      |
      |
      |
      |
      |
      |
      |
      |
      |
      |
    *****
-----|-----|-----|-----
Move a disk. Format: x y

```

As we can see, when we input `1 3` for the first time, the top disk of Rod 1 is popped and pushed to Rod 3.

Then we give an illegal command `1 3` (since it violate the rule: No larger disk may be placed on top of a smaller disk.), it will be ignored, so the previous canvas is printed.

3.1.3 Auto Play Mode

Then we tried the auto-play mode by giving the command `0 0`. As we can see, the program executes automatically from the begining state until the player wins.

```
0 0
Auto moving:3->1  // restore the the beginning state first
```

A 10x10 grid of dots. The top-left 3x3 square of dots is highlighted in light gray. The rest of the grid is white.

Auto moving:1->3

Auto moving:1->2

Auto moving:3->2

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18
19	20	21
22	23	24
25	26	27
28	29	30
31	32	33
34	35	36
37	38	39
40	41	42
43	44	45
46	47	48
49	50	51
52	53	54
55	56	57
58	59	60
61	62	63
64	65	66
67	68	69
70	71	72
73	74	75
76	77	78
79	80	81
82	83	84
85	86	87
88	89	90
91	92	93
94	95	96
97	98	99
100	101	102
103	104	105
106	107	108
109	110	111
112	113	114
115	116	117
118	119	120
121	122	123
124	125	126
127	128	129
130	131	132
133	134	135
136	137	138
139	140	141
142	143	144
145	146	147
148	149	150
151	152	153
154	155	156
157	158	159
160	161	162
163	164	165
166	167	168
169	170	171
172	173	174
175	176	177
178	179	180
181	182	183
184	185	186
187	188	189
190	191	192
193	194	195
196	197	198
199	200	201
202	203	204
205	206	207
208	209	210
211	212	213
214	215	216
217	218	219
220	221	222
223	224	225
226	227	228
229	230	231
232	233	234
235	236	237
238	239	240
241	242	243
244	245	246
247	248	249
250	251	252
253	254	255
256	257	258
259	260	261
262	263	264
265	266	267
268	269	270
271	272	273
274	275	276
277	278	279
280	281	282
283	284	285
286	287	288
289	290	291
292	293	294
295	296	297
298	299	300
301	302	303
304	305	306
307	308	309
310	311	312
313	314	315
316	317	318
319	320	321
322	323	324
325	326	327
328	329	330
331	332	333
334	335	336
337	338	339
340	341	342
343	344	345
346	347	348
349	350	351
352	353	354
355	356	357
358	359	360
361	362	363
364	365	366
367	368	369
370	37	

```

|               |
|               |
|         ***   |
|         |     |
|        *****|
-----|-----|-----|-----
Congratulations! You win!
How many disks do you want? (1 ~ 5)
Q    // quit the game now
```

3.2 Example2

Use normal-mode to win the game yourself.

[illegible]

2 4

1 2

3 2

```
Q // quit the game now
```

4 Submission

Compress your source code into a 7z file, and rename it to `lab3-{your studentid}.7z`

Upload it to canvas.

Feel free to ask teaching assistants if you have any questions.