Creative Software Design, Assignment 7-2

Deadline: 2024-10-20 23:59 (No score for late submission)

- Submit your homework by uploading your zip file to the LMS assignment section. Below is an example.

```
13178_Assignment1-1_2024123456.zip
├─ 1.cc
├─ 2.cc
├─ 3.cc
└─ ...
```

- Your zip file name should follow this format:
  13178_Assignment[Assignment-number]_[Student-ID].zip
  - Ex. 13178_Assignment1-1_2024123456.zip
- Source files should be named as **<filename>.cc** *or* **<filename>.cpp**
- **You must submit your solution in the zip file before the deadline.**

1. **Write a C++ Program to Implement Stack and Queue.**

   A. **Implement `MyStack` and `MyQueue` classes by inheriting the given `Container` class. `MyStack` and `MyQueue` should implement the stack and queue data structures, respectively.**

      1. **Do not use** any STL containers or external libraries to store data.
      2. The data type for `MyStack` and `MyQueue` should be `int`, and the data should be stored in a **dynamic array.**
      3. Implement the constructor, destructor, and the functions `push()`, `pop()`, and `isEmpty()` for both classes.
      4. The maximum size of `MyStack` and `MyQueue` should be set by the constructor.
      5. If `MyStack` or `MyQueue` are full, the dynamic array should be resized to double its current size before adding new elements.
      6. If `MyStack` or `MyQueue` are empty and `pop()` is called, the function should do nothing to prevent underflow.

   B. **You are given the following `Container` class definition, which should not be modified.**

```
class Container {
public:
    Container() {}

    virtual void push(int value) {}
    virtual int pop() { return 0; }
    virtual bool isEmpty() { return false; }
    virtual ~Container() {}
};
```

C. **Example of the `main()` function.**

```
int main() {
    Container *myStack = new MyStack(5);
    Container *myQueue = new MyQueue(5);

    for (int i = 0; i < 10; i++) {
        myStack->push(i * 10);
        myQueue->push(i * 10);
    }

    for (int i = 0; i < 10; i++) {
        int n = myStack->pop();
        cout << n << " ";
    }
    cout << endl;

    for (int i = 0; i < 10; i++) {
        int n = myQueue->pop();
        cout << n << " ";
    }
    cout << "\n";

    delete myStack;
    delete myQueue;

    return 0;
}
```

D. Example output of your program (Bold text indicates user input):

```
90 80 70 60 50 40 30 20 10 0
0 10 20 30 40 50 60 70 80 90
```

E. Submission file: one C++ source file (File name: 1.**cc** or 1.**cpp**)

2. Write a C++ Program for Drawing Multiple 2D Shapes.

A. Requirements

1. The user should first input the size of the canvas (width and height).
2. The program should take user input repeatedly to create objects of the child classes of Shape.
3. Once a shape is created, store it in a `std::vector<Shape*>` and redraw all shapes when displaying the canvas.
4. Shapes must be able to overlap. New shapes should overwrite previously drawn shapes.
5. Empty spaces are printed as **.** (period) and the shape's area is filled with the brush character.

B. The program should accept the following user commands:

1. `add [shape] [parameters]` – This command allows the user to add a new shape to the canvas. The parameters for the shape depend on the specific type of shape being added:
   i. `rect [top-left x] [top-left y] [width] [height] [brush]`
   ii. `diamond [top-center x] [top-center y] [distance from center] [brush]`
   iii. `tri_up [top-center x] [top-center y] [height]`
   iv. `tri_down [bottom-center x] [bottom-center y] [height]`
2. `delete [shape index]` – Deletes the shape at the specified index. If the index does not exist, the program should do nothing.
3. `draw` – Draws the canvas, showing all the shapes that have been added.
4. `dump` – Print information about all shapes.
5. `resize [new width] [new height]` – Resizes the canvas, adjusting its dimensions while trying to preserve existing shapes.
6. `quit` – Quits the program.

C. You are given the following `Canvas` and `Shape` class definition.

1. Complete the required members and functionality (highlighted in blue text).

```
class Canvas {
public:
    Canvas(size_t row, size_t col);
    ~Canvas();

    // Resize the canvas while preserving existing shapes.
    void Resize(size_t w, size_t h);

    // Draw at (x, y) with the given brush.
    bool DrawPixel(int x, int y, char brush);

    // Print the current state of the canvas.
    void Print();

    // Clear the canvas (initialize with '.').
    void Clear();

private:
    // Data members to store drawn shapes.
    // (Make sure to preserve them during resize)
};

class Shape {
public:
    virtual ~Shape();
    virtual void Draw(Canvas* canvas) = 0;

protected:
    // Common properties of shapes
};
```

D. Define the following shape classes by inheriting from the `Shape` class:

1. **`Rectangle`**
2. **`UpTriangle`**
3. **`DownTriangle`**
4. **`Diamond`**

E. Example output of your program (Bold text indicates user input):

```
10 10⏎
 0123456789
0..........
1..........
2..........
3..........
4..........
5..........
6..........
7..........
8..........
9..........

add rect 4 4 3 3 *⏎
draw⏎
 0123456789
0..........
1..........
2..........
3..........
4....***...
5....***...
6....***...
7..........
8..........
9..........

add tri_down 3 3 3 @⏎
draw⏎
 0123456789
0..........
1.@@@@@....
2..@@@.....
3...@......
4....***...
5....***...
6....***...
7..........
8..........
9..........

add tri_up 5 7 3 #⏎
add diamond 2 5 2 ?⏎
draw⏎
 0123456789
0..........
1.@@@@@....
2..@@@.....
3...@......
4....***...
5..?.***...
6.???***...
7?????#....
8.???###...
9..?#####..
```

```
(continue)
add rect 5 5 8 4 +⏎
dump⏎
0 rect 4 4 3 3 *
1 tri_down 3 3 3 @
2 tri_up 7 7 3 #
3 diamond 2 5 2 ?
4 rect 5 5 8 4 +

draw⏎
 0123456789
0..........
1.@@@@@....
2..@@@.....
3...@......
4....***...
5..?.*+++++
6.???*+++++
7?????+++++
8.???#+++++
9..?#####..

delete 5⏎
delete 0⏎
dump⏎
0 tri_down 3 3 3 @
1 tri_up 7 7 3 #
2 diamond 2 5 2 ?
3 rect 5 5 8 4 +

draw⏎
 0123456789
0..........
1.@@@@@....
2..@@@.....
3...@......
4..........
5..?..+++++
6.???.+++++
7?????+++++
8.???#+++++
9..?#####..

resize 15 10⏎
 012345678901234
0...............
1.@@@@@.........
2..@@@..........
3...@...........
4...............
5..?..++++++++..
6.???.++++++++..
7?????++++++++..
8.???#++++++++..
9..?#####.......

quit⏎
```

F. Submission file: one C++ source file (File name: 2.**cc** or 2.**cpp**)