# 과제 #7

HW1

과제 제출 마감: 11/26 24:00

조교 노인우, inwoo13@hanyang.ac.kr

조교 한중수, soohan@hanyang.ac.kr

# Homework

- 과제 제출 마감: 11/26 24:00
- Gitlab repository에 “HW7” 폴더를 만든 후 진행
- 실습 서버 제출, 개인 PC 제출 중 편한 방법으로 제출
- 각 과제마다 HW7 폴더 내에 과제별 폴더를 만든 후 제출
- 채점 기준은 실습서버 환경에서 채점

# Homework_01 – "도형 그리기2 (shape drawing 2)"

- 캔버스의 크기를 입력 받고, 캔버스 내에 도형을 그리는 프로그램 작성
- 다음 클래스를 상속을 이용하여 작성

- class Canvas {

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

        // canvas 크기를 w * h 로 변경한다. 그려진 내용은 보존한다.
        void Resize (size_t w, size_t h)

        // (x, y) 위치에 문자를 그린다. 범위 밖의 x, y는 무시한다.
        bool Draw(int x, int y, char brush);

        // 그려진 내용을 모두 지운다. ('.'으로 초기화)
        void Clear();

private:
        // 그려진 모양을 저장할 수 있도록 데이터멤버를 정의 (resize 가능에 주의)
        friend ostream& operator<<(ostream& os, const Canvas& c);
};
```

# Homework_01 – "도형 그리기2 (shape drawing 2)"

- 캔버스의 크기를 입력 받고, 캔버스 내에 도형을 그리는 프로그램 작성
- 다음 클래스를 상속을 이용하여 작성
- 
```
class Shape {
        public:
                virtual ~Shape();
                virtual void Draw(Canvas* canvas) const = 0;

        protected:
                // 도형의 공통 속성 정의
};

class Rectangle : public Shape { /* 필요한 멤버를 정의 */ }
class UpTriangle : public Shape { /* 필요한 멤버를 정의 */ }
class DownTriangle : public Shape { /* 필요한 멤버를 정의 */ }
class Diamond : public Shape { /* 필요한 멤버를 정의 */ }

istream& operator>>(istream& is, Rectangle& r);
istream& operator>>(istream& is, UpTriangle& t);
istream& operator>>(istream& is, DownTriangle& d);
istream& operator>>(istream& is, Diamond& dm);
```

# Homework_01 – "도형 그리기2 (shape drawing 2)"

- 다음에 유의하여 작성
  - 처음 실행 시 캔버스의 크기를 입력 받음 (가로길이, 세로길이)
  - 모든 도형 정보를 vector에 저장하고 매 출력 시 모두 다시 그리는 형식으로 구성
  - 캔버스 내의 모든 도형은 '겹쳐 그리는 것 ' 이 가능해야 함
    - 뒤에 입력된 도형이 앞에 입력된 도형을 덮어서 그림
  - main() 함수에서 명령, 그리기 원하는 도형의 중심 좌표, 넓이, 높이, 모양(brush) 등을 입력 받음
    - 명령: add, delete, draw, dump, quit // add 함수의 경우 (x좌표, y좌표) 순서로 입력.
    - 도형 종류:

      rect : 좌상단의 x, y 좌표, 넓이, 높이, brush
      tri_up, tri_down : 꼭지점의 x, y 좌표, 높이, brush
      diamond : 상단의 x, y 좌표, 중심-꼭지점간 거리, brush
  - 공백은 '.'으로, 도형 범위 내의 공간은 brush 문자로 출력.
  - main 함수에서 도형의 목록을 vector<Shape*>로 관리하면서 위의 명령을 수행
  - Dump() 함수는 현재 배열에 있는 도형 정보를 index와 함께 표시

- 파일명: draw_shape2
  (draw_shape2.h draw_shape2.cc draw_shape_main2.cc)

# Homework_01 – "도형 그리기2 (shape drawing 2)"

## ◆ 입력 및 출력 예시

```
$ ./draw_shape
10 10
 0123456789
0..........
1..........
2..........
3..........
4..........
5..........
6..........
7..........
8..........
9..........          // 처음 캔버스를 생성하면 빈 캔버스를 1 회 출력
add rect 4 4 3 3 *
draw
 0123456789
0..........
1..........
2..........
3..........
4....***...
5....***...
6....***...
7..........
8..........
9..........          // 좌표 (5, 5)를 좌상단으로 너비, 폭이 3 인 사각형 그림
add tri_down 3 3 3 @
draw
 0123456789
0..........
1.@@@@@....
2..@@@.....
3...@......
4....***...
5....***...
6....***...
7..........
8..........          // 좌표 (3, 3)을 꼭지점으로 하는 높이 3 의 역삼각형
9..........
 0123456789
0..........
1.@@@@@....
2..@@@.....
3...@......
4....***...
5....***...
6....***...
7.....#....
8....###...
9...#####..          // 좌표 (7, 7)을 꼭지점으로 하는 높이 3 의 삼각형
add diamond 2 5 2 ?
draw
 0123456789
0..........
1.@@@@@....
2..@@@.....
3...@......
4....***...
5..?.***...
6.???***...
7??????#....
8.???###...
9..?#####..
// 좌표 (2, 5)를 상단 꼭지점으로 하는 중심에서부터 길이 2 인 다이아몬드
add rect 5 5 8 4 +
// 입력되는 도형의 공간이 캔버스의 범위를 넘어나도 캔버스 내부에 그림.
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 +
```

# Homework_01 – "도형 그리기2 (shape drawing 2)"

## ◆ 입력 및 출력 예시

```
draw
 0123456789
0..........
1.@@@@@....
2..@@@.....
3...@......
4....***...
5..?.*++++++
6.???*++++++
7?????++++++
8.???#++++++
9..?######..
delete 5            // 없는 인덱스를 삭제하는 명령은 무시됨
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        // 캔버스 크기를 수정함
draw
```

```
 012345678901234
0...............
1.@@@@@.........
2..@@@..........
3...@...........
4...............
5..?..++++++++..
6.???.++++++++..
7?????++++++++..
8.???#++++++++..
9..?######.......   // 캔버스 크기 때문에 못 나왔던 도형들 마저 표시
quit
$
```

# Skeleton Code

```
// draw_shape_main.cc

#include <iostream>
#include <string>
#include <vector>
#include "draw_shape.h"

using namespace std;

int main() {
   vector<Shape*> shapes;
   size_t row, col;
   cin >> row >> col;
   Canvas canvas(row, col);
   cout << canvas;
   while (true) {
      string tok;
      cin >> tok;
      if (tok == "add") {
         string type;
         cin >> type;
         if (type == "rect") {
            Rectangle* shape = new Rectangle();
            cin >> *shape;
            shapes.push_back(shape);
         }
         else if (type == "tri_up") {
            UpTriangle* shape= new UpTriangle();
            cin >> *shape;
            shapes.push_back(shape);
         }
```

# Skeleton Code

```
    else if (type == "tri_down") {
        DownTriangle* shape = new DownTriangle();
       cin >> *shape;
       shapes.push_back(shape);
    }
    else if (type == "diamond") {
        Diamond* diamond = new Diamond();
        cin >> *diamond;
        shapes.push_back(diamond);
    }
    else continue;
} else if (tok == "draw") {
    canvas.Clear();
    for (int i = 0; i < shapes.size(); ++i) shapes[i]->Draw(&canvas);
    cout << canvas;
} else if (tok == "delete") {
    int index;
    cin >> index;
    if(index < shapes.size()) shapes.erase(shapes.begin()+index);
} else if (tok == "dump") {
    for(int i=0;i<shapes.size();i++) {
        if(shapes[i]->type()=="rect")
            cout << i << " rect "<< shapes[i]->x() << ' ' << shapes[i]->y() << ' ' << shapes[i]->w() << ' ' << shapes[i]->h() << ' ' << shapes[i]->brush() << endl;
        else
            cout << i <<' ' <<shapes[i]->type() << ' ' <<shapes[i]->x() << ' ' << shapes[i]->y() << ' '  << shapes[i]->h() <<' ' << shapes[i]->brush() << endl;
    }
```

# Skeleton Code

```
        } else if (tok == "resize") {
            int row,col;
            cin >> row >> col;
            canvas.Resize(row,col);
        } else {
            break;
        }
    }
  for (int i = 0; i < shapes.size(); ++i) {
    delete shapes[i];
  }
  shapes.clear();
  return 0;
}
```


Q&A