# 창의적 소프트웨어 설계 실습 문제 09 – hw09-2

**제출 기한**

11월 19일 월 23:59 PM

1. hw09-2(mkdir hw09-2)라는 폴더를 만들고 GitLab에 push
2. hw09-2 디렉토리에 CMakeLists.txt과 각 문제에서 요구하는 파일들을 작성
3. **‘cmake’ 와 ‘make’명령을 수행하여 숙제가 모두 빌드**
4. 최종 버전을 GitLab에 commit
5. **시간과 파일명, 입력과 출력 방식 반드시 지키기**

## 과제 1.

아래의 요구 조건을 만족하고 아래 실행 예와 같이 동작하는 프로그램을 작성하시오.

* Animal class는 두 개의 멤버변수 std::string형 name, int형 age를 가지며, 문자열과 정수를 인자로 받아 name, age의 값을 초기화 하는 생성자를 가진다.
* Animal을 상속받는 Zebra class는 int형 numStripes라는 멤버 변수를 가지며, 문자열, 정수, 정수를 받아 name, age, numStripes의 값을 초기화하는 생성자를 가진다. 이 때, 부모 클래스의 멤버변수는 부모 클래스의 생성자를 통하여 초기화하도록 한다.
* Animal을 상속받는 Cat class는 std::string형 favoriteToy라는 멤버 변수를 가지며, 문자열, 정수, 문자열을 받아 name, age, favoriteToy의 값을 초기화하는 생성자를 가진다. 이 때, 부모 클래스의 멤버변수는 부모 클래스의 생성자를 통하여 초기화하도록 한다.
* Animal, Zebra, Cat class는 각각 printInfo()라는 멤버함수를 가지며, Animal의 경우는 아무 일도 하지 않고, Zebra와 Cat의 경우는 동물의 종류, 이름, 나이, 줄무늬의 개수 (혹은 좋아하는 장난감)을 아래 실행 예의 형식으로 출력한다.
* 사용자부터 ‘z’를 입력 받으면 이후에 입력되는 세 단어는 각각 생성되는 Zebra 객체의 name, age, numStripes를 의미하며, ‘c’를 입력 받으면 이후에 입력되는 세 단어는 각각 생성되는 Cat 객체의 name, age, favoriteToy를 의미한다. ‘0’이 입력되면 입력이 끝났음을 의미한다.
* 사용자로부터 정해지지 않은 개수의 Zebra와 Cat들의 정보를 임의의 순서로 받아, 각 정보마다 생성되는 Zebra 혹은 Cat 객체를 std::vector<Aniaml\*> 타입의 벡터 animals에 저장한다.
* 사용자가 ‘0’을 입력하여 입력이 종료되면, 지금까지 생성된 animals의 각 요소의 정보를 각 요소의 printInfo() 함수를 호출하여 아래 실행 예와 같이 입력된 순서와 동일한 순서로 출력한다.
* 사용이 끝난 animals의 각 요소는 할당받은 메모리를 해제해야 하며, 프로그램 코드 전체에 걸쳐 type casting 연산자는 사용하면 안 된다.

파일명 : Animals (Animals.h Animals.cc Animals\_main.cc)

입력 : 각 객체를 나타내는 종류와 정보 입력

출력 : 입력된 각 객체의 정보 출력

|  |
| --- |
| **$ ./Animals**  **z Tom 2 21**  **z Amy 3 22**  **c Kitty 4 mouse**  **c King 3 tower**  **z Elen 5 30**  **0**  **Zebra, Name: Tom, Age: 2, Number of stripes: 21**  **Zebra, Name: Amy, Age: 3, Number of stripes: 22**  **Cat, Name: Kitty, Age: 4, Favorite toy: mouse**  **Cat, Name: King, Age: 3, Favorite toy: tower**  **Zebra, Name: Elen, Age: 5, Number of stripes: 30**  **$** |

## 과제 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 x h 로 변경한다. 그려진 내용은 보존한다.

void Resize(size\_t w, size\_t h);

// (x,y) 위치에 ch 문자를 그린다. 범위 밖의 x,y 는 무시한다.

bool DrawPixel(int x, int y, char brush);

//캔버스의 내용을 화면에 출력한다.

Void Print();

// 그려진 내용을 모두 지운다 ('.'으로 초기화)

void Clear();

};

class Shape {

public:

virtual ~Shape();

virtual void Draw(Canvas\* canvas) {};

protected:

// 도형의 공통 속성을 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 \*/ }

다음에 유의하여 작성

* 처음 실행시 캔버스의 크기를 입력받음.
* 모든 도형 정보를 vector에 저장하고 매 출력시 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, resize
  + 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 명령은 현재 배열에 있는 도형 정보를 index와 함께 표시.

파일명 : draw\_shape (draw\_shape.h draw\_shape.cc draw\_shape\_main.cc)

입력 / 출력

|  |
| --- |
| **$ ./draw\_shape**  **10 10**  **0123456789**  **0..........**  **1..........**  **2..........**  **3..........**  **4..........**  **5..........**  **6..........**  **7..........**  **8..........**  **9.......... // 처음 캔버스를 생성하면 빈 캔버스를 1회 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 3인 사각형 그림**  **add tri\_down 3 3 3 @**  **draw**  **0123456789**  **0..........**  **1.@@@@@....**  **2..@@@.....**  **3...@......**  **4....\*\*\*...**  **5....\*\*\*...**  **6....\*\*\*...**  **7..........**  **8.......... // 좌표 (3, 3)을 꼭지점으로하는 높이 3의 역삼각형**  **9..........**  **add tri\_up 5 7 3 #**  **draw**  **0123456789**  **0..........**  **1.@@@@@....**  **2..@@@.....**  **3...@......**  **4....\*\*\*...**  **5....\*\*\*...**  **6....\*\*\*...**  **7.....#....**  **8....###...**  **9...#####.. // 좌표 (7, 7)을 꼭지점으로하는 높이 3의 삼각형**  **add diamond 2 5 2 ?**  **draw**  **0123456789**  **0..........**  **1.@@@@@....**  **2..@@@.....**  **3...@......**  **4....\*\*\*...**  **5..?.\*\*\*...**  **6.???\*\*\*...**  **7?????#....**  **8.???###...**  **9..?#####.. // 좌표 (2, 5)를 상단 꼭지점으로하는 중심에서부터 길이 2인 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 +**  **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**  **$** |
|  |

Code skeleton

|  |
| --- |
| // 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);  canvas.Print();  while (true) {  string tok;  cin >> tok;  if (tok == "add") {  string type;  cin >> type;  if (type == "rect") {  // 이 부분은 직접 작성  }  else if (type == "tri\_up") {  // 이 부분은 직접 작성  }  else if (type == "tri\_down") {  // 이 부분은 직접 작성  }  else if (type == "diamond") {  // 이 부분은 직접 작성  }  else continue;  }  else if (tok == "draw") {  canvas.Clear();  for (int i = 0; i < shapes.size(); ++i) shapes[i]->Draw(&canvas);  canvas.Print()  }  else if (tok == "delete") {  // 이 부분은 직접 작성  }  else if (tok == "dump") {  // 이 부분은 직접 작성  }  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;  } |