Anh Nguyen

Professor Chia Yung Han

Data Structure

11 December 2019

Midterm Redo

**Problem 1**:

1. We define this class before int main():

//1. TicketInfo Structure

struct TicketInfo{

int TicketNumber;

double price;

string SeatingInfo;

};

1. Snipper of code for Seat structure for inputting Ticket Sales information:

- Since the number of columns and rows for each section are constant, we can put them as global variables outside of int main():

int ROW = 16;

int COL = 20;

- We define such array in int main():

TicketInfo Seat[ROW][COL];

- We put each element in the loop for both rows and columns, to input information for each seat as followed:

for (int i = 0; i < ROW ; i++){

for (int j = 0; j < COL; i++){

cout << "Enter Ticket number: "<< endl;

cin >> Seat[i][j].TicketNumber;

cout << "Enter price: "<< endl;

cin >> Seat[i][j].price;

cout << "Enter Seating information: " << endl;

cin >> Seat[i][j].SeatingInfo;

}

}

1. (No points missed)
2. When passing a 2D array through a function, the array passed must always include its second size declarator for the column:

TicketInfo ArrayFunction(TicketInfo Arr[][COL]){

return Arr[0][0];

}

This is legal:

TicketInfo ArrayFunction(TicketInfo Arr[ROW][COL])

These are not:

TicketInfo ArrayFunction(TicketInfo Arr[][]){}

TicketInfo ArrayFunction(TicketInfo Arr[ROW][]){}

**Problem 2**:

1. (No points missed)
2. We can define a dynamic objects using a pointer to the class:

Frame\* viewGraphics;

viewGraphics = new Frame("View graphics", 5, 2, 6, 3);

viewGraphics->Resize(200, 150);

1. \*It may be very beneficial to build a default constructor, it’ll be in Public inside the class Frame:

Frame(){

label = "";

x\_coord = 0;

y\_coord = 0;

width = 0;

height = 0;

}

- We’re using class Frame to initialize an array of 5 default frame:

Frame winList[5];

- We’re using class Frame to initialize an array of 5 default frame:

for (int i = 0; i<5; i++){

winList[i].Resize(20,20);

}

-Increment x-coordinate by 20 units for every Frame. Keep their y-coordiate equal to each other, since we want them next to each other:

int incrementX = 0;

for (int i = 0; i<5; i++){

winList[i].MoveTo(incrementX, 0);

incrementX = incrementX + 20;

}

**Problem 3**:

- To create the MoveTo() and Resize() functions, we need to add some set functions to class Loc and Shape:

void Shape::setWidth(int w){

width = w;}

void Shape::setHeight(int h){

height = h;}

void Loc::setY(int y){

y\_coord = y;}

void Loc::setX(int x){

x\_coord = x;}

- Class Frame will inherit both Shape and Loc, which will grant it access to all publics functions from both classes:

class Frame : public Shape, public Loc{

private:

string label;

public:

Frame(string name, int X, int Y, int width, int height) : Shape(width, height), Loc(X,Y){

label = name;

}

void Resize(int w, int h){

setHeight(h);

setWidth(w);

}

void Moveto(int x, int y){

setX(x);

setY(y);

}

};

**Problem 4**:

a)

- Abstract base class:

class animal{

private:

int x;

public:

//These 2 functions are to be overriden in the derived class

virtual int getAge() = 0;

virtual void show() = 0;

int showX(){

return x;

}

};

- Derived class:

class mammal : public animal{

private:

int y;

public:

//Overriding the virtual functions defined in animal

int getAge(){

return y;

}

void show(){

cout << showX() << endl;

cout << y << endl;

}

};

b) (No points missed)

**Problem 5**: Queue and Vectors