Project 1

The Card Matching Game

CSC-48096

Name: Matthew Yang

Date 10/28/16

Introduction

Title: Matching Card Game

This is a card matching game.

The cards are set before you, and the user needs to input two cards one at a time in order to reveal what number is underneath those two cards. The goal of the game is to pick two cards that have the same number underneath them.

When all cards are matched successfully, the game is over.

This is a fun family memory game that many play to see how fast they can finish the game in the least amount of turns.

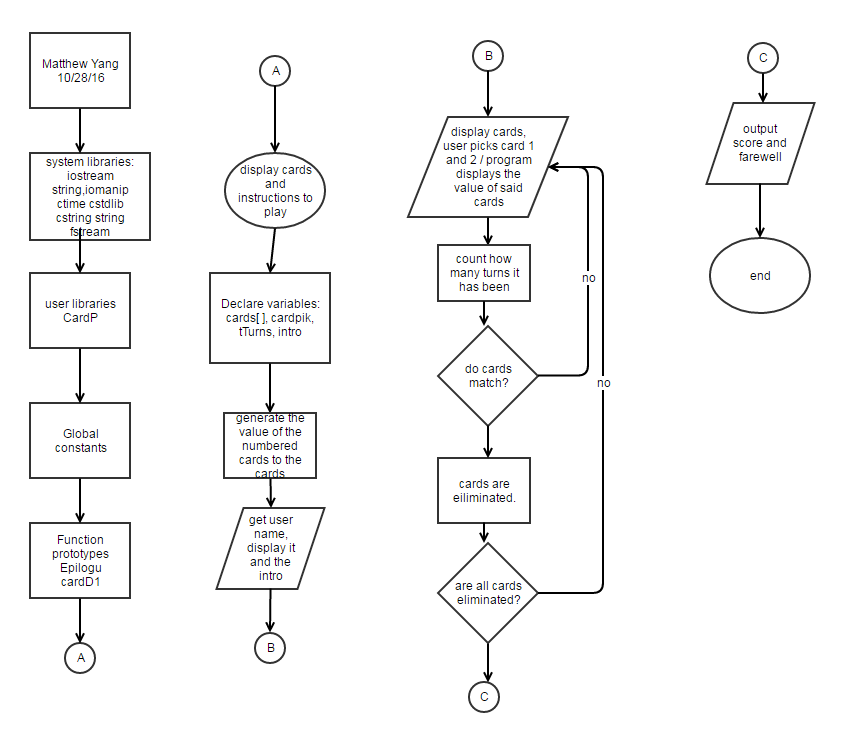
Summary

Project size: 298 lines

Variables: there are 3 main variables that aren’t in an array, and there are 2 main arrays with one having 26 variables, the second containing 13 variables. There are also 2 in a structure. Therefore, there are 44 main variables in all.

This project contains most concepts we have learned in class. I feel good about how I finally figured out why I kept on failing at my binary file. I could have introduced points into my game to improve competitiveness. It took me what felt like forever.

Flowchart



Major variables

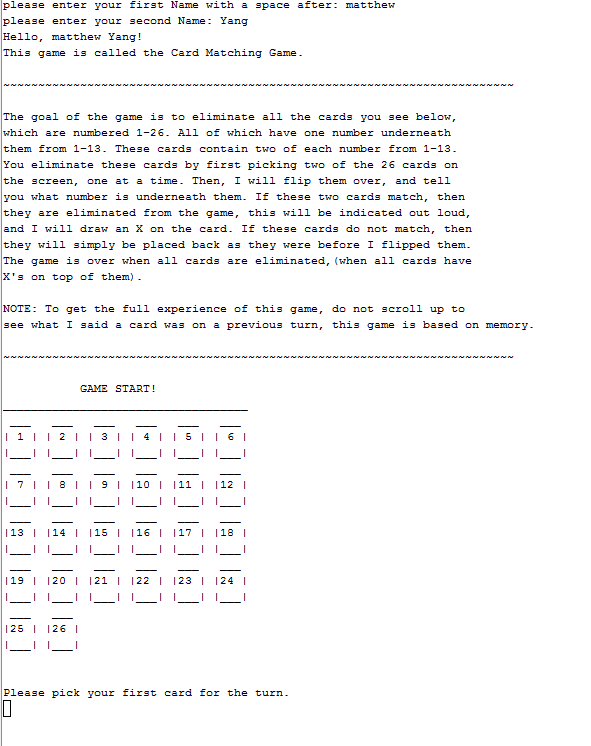
Type Variable Name Description Location

|  |  |  |  |
| --- | --- | --- | --- |
| integer | SIZE | Number of cards (26) | In declared variables |
|  | tTurns | How many turns has passed | In the epilogu(); function |
| Integer arrays | cards[ ] | Value in each card 1-13 | After the declaration of variables, and when the program reads the revealed values back to the user |
|  | freqofc[ ] | Used to make sure each card is picked only twice | After the declaration of variables |
|  | cardpik | User input to choose cards | Inside my CardP structure |
| bool | isDone | Determines when to finish the game | At the end of main, inside the last for-loop |

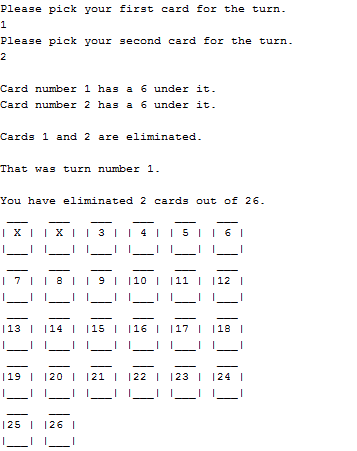
Constructs

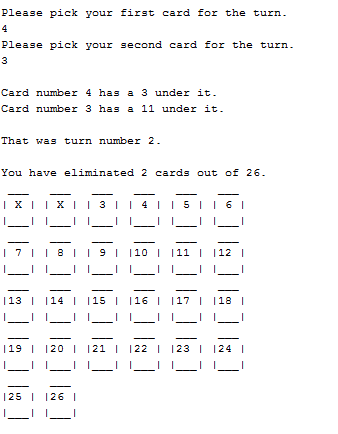
Chapter Keywords Location

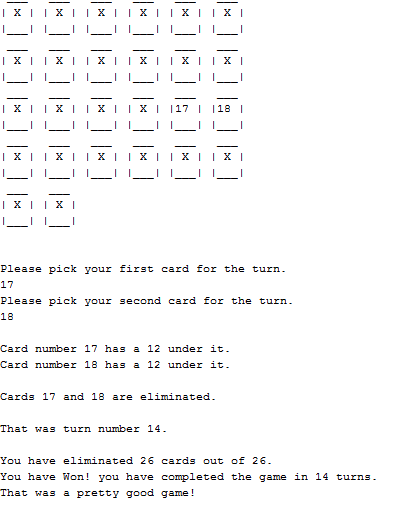
|  |  |  |
| --- | --- | --- |
| Chapter 9 | * Pointer usage for Dynamic Memory Allocation * Initializing pointers | * Declare variables section * After the instruction portion |
| Chapter 10 | * Character arrays * Library functions for working with C-strings * strings | * First two are before the instructions portion * First one is after the character arrays * In the declare variable portion |
| Chapter 11 | * Structure * Mutating structure * Outputting structure | * in the header * when I ask for the card picks * when the game ends, and when at the end of the turn. |
| Chapter 12 | * writing to binary file * reading from binary file | * after the instruction portion * after writing to binary file |

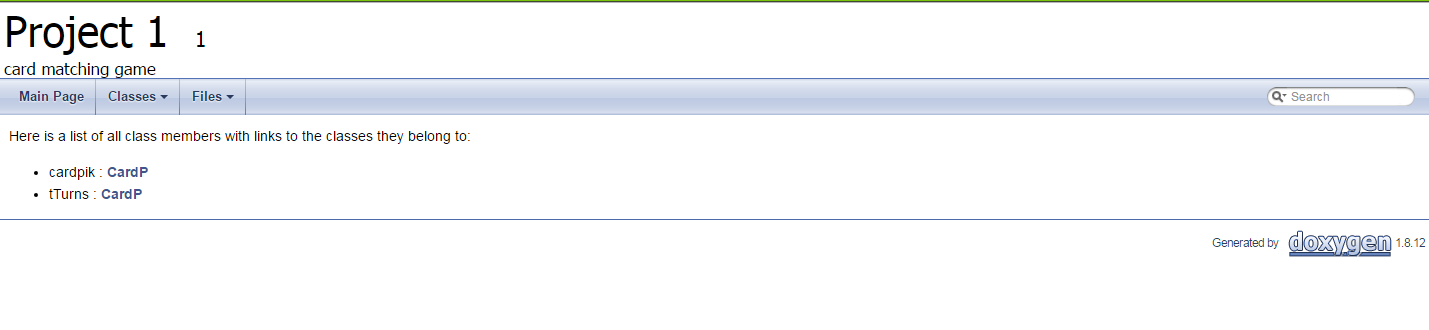
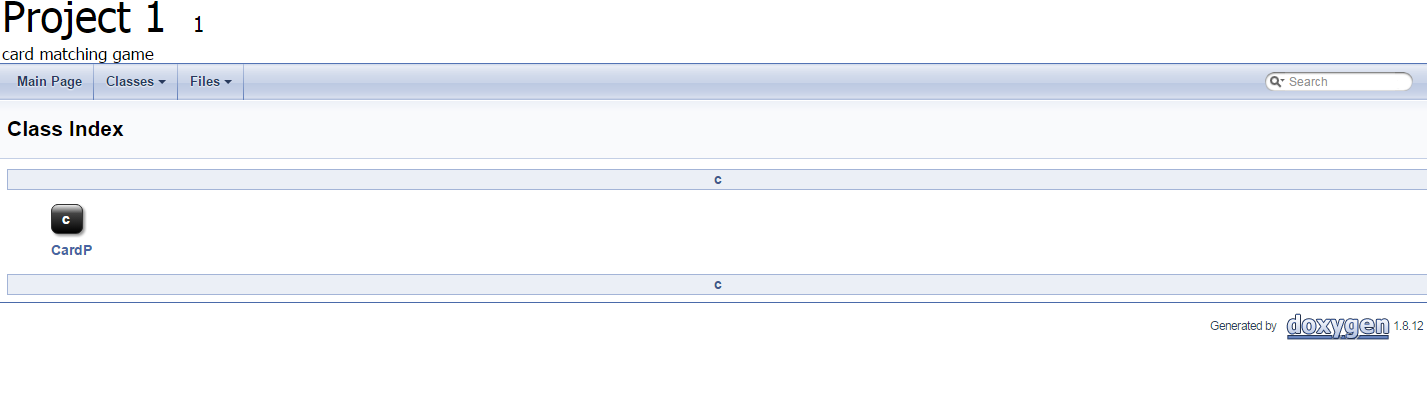
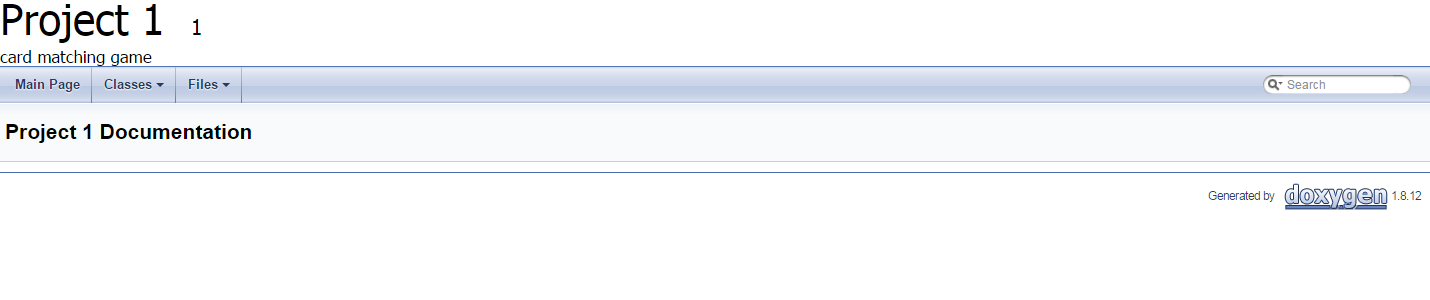
Samples

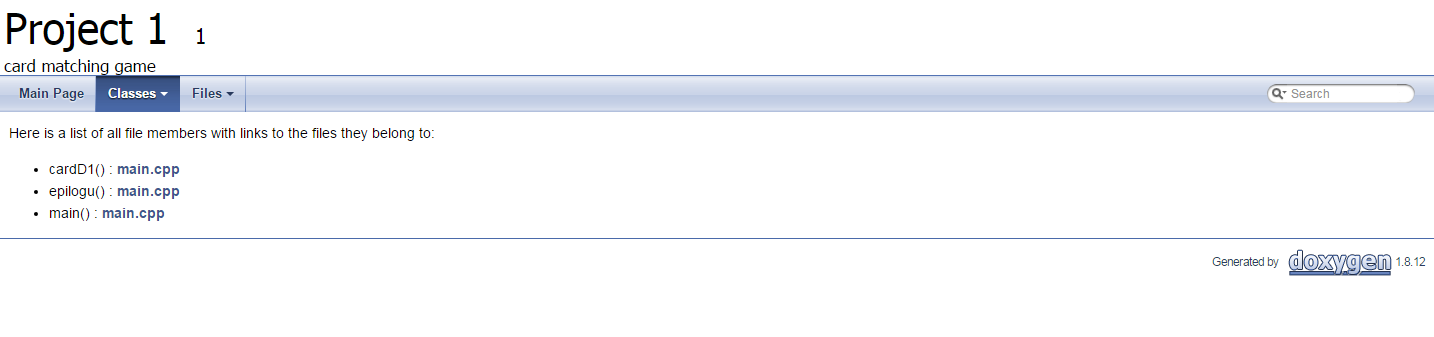
🡨This is just the intro screen with the displayed cards. This screen asks for your name and tells you how to play. The second part of the screen shows you the cards that are on the board.

🡨 this is what it looks like when you successfully eliminate two cards.

 🡨this is what happens when you don’t eliminate 2 cards-nothing. It says I eliminated 2 cards so far, so that doesn’t mean I eliminated them this turn, it was from a previous turn.

🡨 this is what happens when you complete the game, it congratulates you based on how well you did and states the turns that you made.





These are my Doxygen pages that we were told to make.

Pseudocode

*creates randomness*

*create values for each card, while making sure each card is repeated only twice*

*makes sure that no card has the same value more than twice.*

*gets full name*

*outputs the intro and rules to the game into a binary file*

*reads them from the file*

*displays virtual cards while marking X's on them if they are eliminated*

*pick card 1, and makes sure it is a valid number, (1-26 and not eliminated already)*

*pick card 2, and make sure it is a valid number.*

*makes sure that the cards you inputted are different.*

*outputs the results of the cards*

*counts the turn you are on*

*if the cards have the same number underneath them, they are eliminated*

*sets card picks back to zero*

*displays turn*

*counts the number of eliminated cards*

*displays the number of terminated cards*

*loops back to pick cards until you have eliminated all cards*

*finishes the program if 26 cards are eliminated, while displaying the total turns it took you to finish the game*

*tells you how well you did on the game*

Program

//declare variables

bool isDone = false; //determines when game is done

const unsigned int SIZE =26; //total #of cards

int cards[SIZE] = {0}; //total card's values

int freqofC[13] = {0}; //frequency of cards

CardP card[2]; //for the user's pick of cards, 1st and 2nd

unsigned int turns=0; //turns used

string intro;

char \*intrArr;

fstream introPr; //declares prologue file

char line[1200];

unsigned seed = time(0); //creates randomness

srand(seed);

//create values for each card,

//while making sure each card is repeated only twice

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

int num = rand() % 13 + 1;

while (freqofC[num-1]>=2) { //makes sure that no card has the

num = rand() % 13 + 1; //same value more than twice.

}

++freqofC[num-1];

cards[i] = num;

}

const int nameF = 26; //used for name total length of full name

const int nameS = 13; //used for second name length

char fName[nameF]; //first name

char sName[nameS]; //second name

//gets full name

cout <<"please enter your first Name with a space after: ";

cin.getline(fName,nameS);

cout <<"please enter your second Name: ";

cin.getline(sName,nameS);

strcat(fName,sName);

introPr.open("introInstructions.dat", ios::out | ios::binary);

cout << "Hello, "<<fName<<"!"<<endl;

intro= "This game is called the Card Matching Game. \n\n"

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

"~~~~~~~~~~~~\n\n"

"The goal of the game is to eliminate all the cards"

" you see below,\n"

"which are numbered 1-26. All of which have one number underneath \n"

"them from 1-13. These cards contain two of each number from 1-13.\n"

"You eliminate these cards by first picking two of the 26 cards on\n"

"the screen, one at a time. Then, I will flip them over, and tell\n"

"you what number is underneath them. If these two cards match, "

"then\n"

"they are eliminated from the game, "

"this will be indicated out loud,\n"

"and I will draw an X on the card. If these cards do not match,"

" then\n"

"they will simply be placed back as they were before I flipped "

"them.\n"

"The game is over when all cards are eliminated,(when all cards "

"have\n"

"X's on top of them).\n\n"

"NOTE: To get the full experience of this game, do not scroll up to\n"

"see what I said a card was on a previous turn, "

"this game is based on memory.\n\n"

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n"

" GAME START!\n"

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

//outputs the intro and rules to the game into a binary file

intrArr = new char[1200];

strcpy(intrArr,intro.c\_str());

introPr.write(intrArr,1200);

introPr.close();

//reads them from the file

introPr.open("introInstructions.dat", ios::in | ios::binary);

introPr.seekg(0, ios::beg);

introPr.read(line,1200);

cout <<line;

introPr.close();

//loops until game is over

do{

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

cout<<i+1<<". "<<cards[i]<<" ";

}cout<< endl;

cardD1(cards, SIZE); //displays virtual cards while marking X's on

//them if they are eliminated

cout<< endl;

//pick card 1, and makes sure it is a valid number,

//(1-26 and not eliminated already)

cout <<endl<< "Please pick your first card for the turn."<<endl;

cin >> card[0].cardpik;

while(cards[card[0].cardpik-1]==0){

cout << "You have already picked that card. Please pick a card"

" that is on the board, and that does not have an X on it.";

cin >> card[0].cardpik;

}

while (card[0].cardpik<1||card[0].cardpik>SIZE) {cout << "you must pick a card "

"that exists on the screen above, and that does not have an X "

"on it."<<endl;

cin >>card[0].cardpik;

}

//pick card 2, and make sure it is a valid number.

cout << "Please pick your second card for the turn."<<endl;

cin >> card[1].cardpik;

while(cards[card[1].cardpik-1]==0){

cout << "You have already picked that card. Please pick a card"

" that is on the board, and that does not have an X on it.";

cin >> card[1].cardpik;

}

while (card[1].cardpik<1||card[1].cardpik>SIZE) {cout << "you must pick a card "

"that exists on the screen above, and that does not have an X "

"on it."<<endl;

cin >>card[1].cardpik;

}

//makes sure that the cards you inputted are different.

while (card[0].cardpik==card[1].cardpik){

cout << "you must pick a number that is different than the first "

<<endl<<"card you have picked. Pick another card for your"

" second card."<<endl;

cin >>card[1].cardpik;

}

//outputs the results of the cards

cout << endl << "Card number "<< card[0].cardpik<<" has a "<<

cards[card[0].cardpik-1]<<" under it." <<endl;

cout <<"Card number "<< card[1].cardpik<<" has a "<<

cards[card[1].cardpik-1]<<" under it." <<endl<<endl;

turns++; //counts the turn you are on

//if the cards have the same number underneath them,

//they are eliminated

if (cards[card[0].cardpik-1]==cards[card[1].cardpik-1]){ cout <<"Cards "<<

card[0].cardpik<< " and " <<card[1].cardpik <<" are eliminated."

<< endl<<endl;

cards[card[0].cardpik-1]=0; //sets card pick back to zero

cards[card[1].cardpik-1]=0;

}

//displays turn

cout << "That was turn number "<<turns<<"."<<endl<<endl;

int cnt=0; //counts the number of eliminated cards

for(int i=0; i<SIZE; ++i)

if( cards[i]==0 ) ++cnt;

if(cnt==26) isDone = true;

// displays the number of terminated cards

cout << "You have eliminated " << cnt << " cards out of 26." << endl;

}while(!isDone); //finishes the program if 26 cards are eliminated,

epilogu(turns,fName);

delete []card;

//Exit stage right!

return 0;

}

void epilogu(int turns, char fName[]){

CardP card;

card.tTurns=turns;

cout << "You have Won! you have completed the game in " <<card.tTurns<<

" turns.";

cout << endl;

//gives special message if you had a perfect game

switch (turns){

case 13: cout << "WOW! A perfect game! You're amazing!!";

default: cout << "";

}

//tells you how well you did on the game if you did not have

//a perfect game

if (turns>13 && turns<=17){

cout <<"That was a pretty good game!";

}

if (turns>=18 && turns<22){

cout << "That was an OK game, although you can do better.";

}

if (turns>=22 && turns<=25){

cout << "Not a really good game. Try again another time?";

}

if (turns>25){

cout << "That was a bad game.";

}

cout << endl;

cout << "Thanks for playing, "<<fName;

}

CardP cardD1(int cards[], int SIZE) {

string cad1P2= "| 1 |",cad2P2= "| 2 |",

cad3P2= "| 3 |",cad4P2= "| 4 |",

cad5P2= "| 5 |",cad6P2= "| 6 |",

cad7P2= "| 7 |",cad8P2= "| 8 |",

cad9P2= "| 9 |",cad10P2="|10 |",

cad11P2="|11 |",cad12P2="|12 |",

cad13P2="|13 |",cad14P2="|14 |",

cad15P2="|15 |",cad16P2="|16 |",

cad17P2="|17 |",cad18P2="|18 |",

cad19P2="|19 |",cad20P2="|20 |",

cad21P2="|21 |",cad22P2="|22 |",

cad23P2="|23 |",cad24P2="|24 |",

cad25P2="|25 |",cad26P2="|26 |",

cadP1="\_\_\_", cadP3= "|\_\_\_|",

X="| X |";

CardP cardP;

cardP.cardpik =0;

//cards being turned into X's if they are removed from the game

if (cards[0]==0){cad1P2=X;} if (cards[1]==0){cad2P2=X;}

if (cards[2]==0){cad3P2=X;} if (cards[3]==0){cad4P2=X;}

if (cards[4]==0){cad5P2=X;} if (cards[5]==0){cad6P2=X;}

if (cards[6]==0){cad7P2=X;} if (cards[7]==0){cad8P2=X;}

if (cards[8]==0){cad9P2=X;} if (cards[9]==0){cad10P2=X;}

if (cards[10]==0){cad11P2=X;} if (cards[11]==0){cad12P2=X;}

if (cards[12]==0){cad13P2=X;} if (cards[13]==0){cad14P2=X;}

if (cards[14]==0){cad15P2=X;} if (cards[15]==0){cad16P2=X;}

if (cards[16]==0){cad17P2=X;} if (cards[17]==0){cad18P2=X;}

if (cards[18]==0){cad19P2=X;} if (cards[19]==0){cad20P2=X;}

if (cards[20]==0){cad21P2=X;} if (cards[21]==0){cad22P2=X;}

if (cards[22]==0){cad23P2=X;} if (cards[23]==0){cad24P2=X;}

if (cards[24]==0){cad25P2=X;} if (cards[25]==0){cad26P2=X;}

//cards being displayed

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad1P2<<" "<<cad2P2<<" "<<cad3P2<<" "<<cad4P2<<" "<<cad5P2<<" "

<<cad6P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad7P2<<" "<<cad8P2<<" "<<cad9P2<<" "<<cad10P2<<" "<<cad11P2<<" "<<

cad12P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<

endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad13P2<<" "<<cad14P2<<" "<<cad15P2<<" "<<cad16P2<<" "<<cad17P2<<" "

<<cad18P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)<<cadP1<<setw(6)

<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad19P2<<" "<<cad20P2<<" "<<cad21P2<<" "<<cad22P2<<" "<<cad23P2<<" "

<<cad24P2<<endl;

cout <<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3<<" "<<cadP3

<<endl;

cout << " "<<cadP1<<setw(6)<<cadP1<<endl;

cout <<cad25P2<<" "<<cad26P2<<endl;

cout <<cadP3<<" "<<cadP3<<endl;

}