# ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my computer teacher Mrs. Bhargavi Rajeev as well as our principal Mrs. Sumitra Das, of National Public School, who gave me the golden opportunity to work on this wonderful project.

Their help, guidance, support and encouragement has enabled me to complete this protect successfully.

This project has allowed me to do a lot of Research and I have learnt many new things and am really thankful for that.

Secondly i would also like to thank my parents who helped me complete this project within the limited time frame.

# INTRODUCTION

# ABOUT C++

C++ is a middle-level programming language developed by Bjarne Stroustrup starting in 1979 at Bell Labs in Murray Hill, New Jersey, as an enhancement to the C language and originally named C with Classes but later it was renamed C++ in 1983. C++ runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

## C++ - Overview

C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural, object-oriented, and generic programming.

C++ is regarded as a middle-level language, as it comprises a combination of both high-level and low-level language features.

C++ is a superset of C, and that virtually any legal C program is a legal C++ program.

A programming language is said to use static typing when type checking is performed during compile-time as opposed to run-time.

## Object-Oriented Programming

C++ fully supports object-oriented programming, including the four pillars of object-oriented development:

* Encapsulation
* Data hiding
* Inheritance
* Polymorphism

## Standard Libraries

Standard C++ consists of three important parts:

* The core language giving all the building blocks including variables, data types and literals, etc.
* The C++ Standard Library giving a rich set of functions manipulating files strings, etc.
* The Standard Template Library (STL) giving a rich set of methods manipulating data structures, etc.

## The ANSI Standard

The ANSI standard is an attempt to ensure that C++ is portable i.e. for example- a code written for Microsoft's compiler will compile without errors, using a compiler on a Mac, UNIX, a Windows box, or an Alpha.

The ANSI standard has been stable for a while, and all the major C++ compiler manufacturers support the ANSI standard.

## Use of C++

C++ is used by hundreds of thousands of programmers in essentially every application domain.

C++ is being highly used to write device drivers and other softwares that rely on direct manipulation of hardware under real-time constraints.

C++ is widely used for teaching and research because it is clean enough for successful teaching of basic concepts.

The primary user interfaces of either an Apple Macintosh or a PC running Windows are written in C++.

# ABOUT THE PROJECT-

This C++ project is designed to implement a game called PACMAN. This is a simple arcade type game. PACMAN is basically a character controlled by the user, by using the keyboard. The player guides PACMAN through a maze, while eating dots (called pac-dots) which adds to the score. When PACMAN eats enough pac-dots in a particular level, he is promoted to the next level.

A few enemies (called MONSTER) roam the maze, trying to catch Pac-Man. If an enemy touches PACMAN he withers and the game ends.

This game supports multiple players. Players can create accounts, and play independently. Each player’s progress is saved in a dat file in binary format, thus implementing file handling. The player can continue playing from where he last played by just logging in, using his username and password.

PACMAN, the MONSTERS and the player are treated as objects each with characteristic features, working independently and are integrated as a whole, thus enforcing Object Oriented Programming.

The game uses graphics to make the game-play more attractive and user friendly.

## About graphics.h

This interface provides access to a simple graphics library that makes it possible to draw lines, rectangles, ovals, arcs, polygons, images, and strings on a graphical window.

C graphics using graphics.h functions can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h in turbo c compiler one can make graphics programs. One can draw circles, lines, rectangles, bars and many other geometrical figures. He can change their colors using the available functions and fill them. All the functions are defined in the graphics.h header file.

Few functions from graphics.h are listed below:-

* ***int*** *initwindow(****int*** *width,* ***int*** *height,* ***const******char****\* title="Windows BGI");*

The function initializes the graphics system by opening a graphics window.

* ***void*** *fillellipse(****int*** *x****, int*** *y****, int*** *xradius****, int*** *yradius);*

Draws an ellipse using (x,y) as a center point and xradius and yradius as the horizontal and vertical axes.

* ***void*** *sector****(int*** *x****, int*** *y****, int*** *stangle****, int*** *endangle****, int*** *xradius****, int*** *yradius);*

Draws and fills an elliptical pie slice using (x,y) as the center point, xradius and yradius as the horizontal and vertical radii, respectively, and drawing from start angle (stangle) to end angle (endangle).

* ***void*** *bar(****int*** *left****, int*** *top****, int*** *right****, int*** *bottom);*

Draws a filled-in, rectangular, two-dimensional bar. The bar is filled using the current fill pattern and fill colour.

* ***int*** *getmaxx(****void****);*

Returns the maximum (screen-relative) x value for the current graphics state.

* ***int*** *getmaxy(****void****);*

Returns the maximum (screen-relative) y value for the current graphics state.

* ***void*** *getimage(****int*** *left****, int*** *top****, int*** *right****, int*** *bottom****, void*** *\*bitmap);*

Copies an image from the screen to memory.

* ***void*** *putimage(****int*** *left****, int*** *top****, void \*****bitmap****, int*** *op);*

Puts the bit image previously saved with getimage back onto the screen.

* ***void*** *writeimagefile(*

***const char\**** *title=NULL,*

***double*** *width\_inches=7,* ***double*** *border\_left\_inches=0.75,* ***double*** *border\_top\_inches=0.75,*

***int*** *left=0,* ***int*** *right=0,* ***int*** *right=INT\_MAX,* ***int*** *bottom=INT\_MAX*

*);*

The function saves a portion of the active page in a BMP file. The filename must end in "BMP" or ".bmp".

* ***void*** *readimagefile(*

***const char\**** *title=NULL,*

***int*** *left=0,* ***int*** *right=0,* ***int*** *right=INT\_MAX,* ***int*** *bottom=INT\_MAX*

*);*

The function reads a BMP, GIF, JPG, ICO, EMF or WMF image file and displays it in part of the current active window.

* ***unsigned*** *getpixel(****int*** *x,* ***int*** *y);*

Gets the colour of the pixel located at (x,y).

# HEADER FILES USED-

* iostream.h
* string.h
* graphics.h
* conio.h
* fstream.h
* stdlib.h
* time.h
* math.h

**Header Files Created: (**For the Purpose of MODULARITY**)-**

* object.h – contains definition of OBJECT class
* pacman.h – contains definition of PACMAN class
* monster.h – contains definition of MONSTER class
* player.h – contains definition of PLAYER class

# CLASSES-

* **OBJECT**- A base class containing the basic characteristics of a game object.
* **PACMAN**- This inherits the class OBJECT. It defines how the pacman behaves during game execution.
* **MONSTER**- This also inherits the class OBJECT. It defines how the monster behaves during game execution.
* **PLAYER**- This class contains data members and functions to store a player’s progress in the game.

# FUNCTIONS-

## MEMBER FUNCTIONS:

* class **OBJECT**:-
  + void initialize();
    - It is a private member.
    - No implementation details included. Implementation details can be provided in the sub classes.
  + int getCOX();
    - It is a public member.
    - Returns the x coordinate of the object.
    - provided in the sub classes.
  + int getCOY();
    - It is a public member.
    - Returns the y coordinate of the object.
  + void step();
    - It is a public member.
    - No implementation details included. Implementation details can be provided in the sub classes.
* class **PACMAN**:- This class inherits the class OBJECT in public mode.
  + void initialize();
    - It is a private member.
    - It creates the image of Pacman on the screen, using functions defined in graphics.h.
    - It sets pacman’s initial position on the screen.
    - It initializes other data members.
  + void PACMAN();
    - It is the default constructor of the class.
    - It is a public member.
    - It calls the member function initialize().
  + void PACMAN(int x , int y);
    - It is the parameterized constructor of the class.
    - It is a public member.
    - It takes the initial position coordinates of Pacman as the parameters.
    - It creates Pacman at the desired position by calling the member function initialize().
  + void clearPacman();
    - It is a public member.
    - It clears the image of Pacman from the screen.
  + void erasePacman();
    - It is a public member.
    - It also clears the image of Pacman from the screen accompanied with some animation.
  + void ~PACMAN();
    - It is the destructor of the class.
    - It is a public member.
    - It calls the member function erasePacman().
  + int obstacleUp();
    - It is a public member.
    - It checks for obstacles above Pacman by analysing the colour of the pixels above it.
    - It returns 0 if there are no obstacles and returns the colour if there is an obstacle.
  + int obstacleDown();
    - It is a public member.
    - It checks for obstacles below Pacman by analysing the colour of the pixels above it.
    - It returns 0 if there are no obstacles and returns the colour if there is an obstacle.
  + int obstacleLeft();
    - It is a public member.
    - It checks for obstacles to the left of Pacman by analysing the colour of the pixels to its left.
    - It returns 0 if there are no obstacles and returns the colour if there is an obstacle.
  + int obstacleRight();
    - It is a public member.
    - It checks for obstacles to the right of Pacman by analysing the colour of the pixels to its right.
    - It returns 0 if there are no obstacles and returns the colour if there is an obstacle.
  + void stepType1();
    - It is a public member.
    - This is responsible for moving Pacman across the screen. It moves pacman by only 1 step.
    - It only moves Pacman if no obstacle is present in its way.
    - It takes input from the user in order to steer Pacman, however doesn’t wait for a key to be pressed.
    - If no key is pressed, it continues to make Pacman move in the direction indicated by the last key pressed, until another key is pressed.
* class **MONSTER**:- This class inherits the class OBJECT in public mode. It is also a Self Referential class.
  + void initialize();
    - It is a private member.
    - It creates the image of a Monster on the screen, using functions defined in graphics.h.
    - It sets the monster’s initial position on the screen.
    - It initializes other data members.
  + void MONSTER();
    - It is the default constructor of the class.
    - It is a public member.
    - It calls the member function initialize().
  + void MONSTER(int x , int y);
    - It is the parameterized constructor of the class.
    - It is a public member.
    - It takes the initial position coordinates of Pacman as the parameters.
    - It creates Pacman at the desired position by calling the member function initialize().
  + void clearMonster();
    - It is a public member.
    - It clears the image of Pacman from the screen.
  + void ~MONSTER();
    - It is the destructor of the class.
    - It is a public member.
    - It calls the member function clearMonster ().
  + int obstacleUp();
    - It is a public member.
    - It checks for obstacles above the Monster by analysing the colour of the pixels above it.
    - It returns 0 if there are no obstacles and returns the colour if there is an obstacle.
  + int obstacleDown();
    - It is a public member.
    - It checks for obstacles below the Monster by analysing the colour of the pixels above it.
    - It returns 0 if there are no obstacles and returns the colour if there is an obstacle.
  + int obstacleLeft();
    - It is a public member.
    - It checks for obstacles to the left of the Monster by analysing the colour of the pixels to its left.
    - It returns 0 if there are no obstacles and returns the colour if there is an obstacle.
  + int obstacleRight();
    - It is a public member.
    - It checks for obstacles to the right of the Monster by analysing the colour of the pixels to its right.
    - It returns 0 if there are no obstacles and returns the colour if there is an obstacle.
  + void Step ();
    - It is a public member.
    - This is responsible for moving the Monster across the screen. It moves the Monster by only 1 step.
    - It only moves Pacman if no obstacle is present in its way.
    - It moves the Monster in a direction determined randomly, at each step.
* class **PLAYER**:-
  + void PLAYER();
    - It is the default constructor of the class.
    - It is a public member.
    - It initializes the data members with their default values.
  + void create();
    - It is a public member.
    - It is used to create a new account for the player.
    - It accepts the player’s name and password.
  + void login();
    - It is a public member.
    - It is used while logging into the player’s account.
  + int TotalScore();
    - It is a public member.
    - It returns the sum of all scores obtained in each level.
  + void display();
    - It is a public member.
    - It displays player’s details.
  + void incrementLevel();
    - It is a public member.
    - It increases the player’s current level by 1.
  + void setScore(int level);
    - It is a public member.
    - It accepts the level the player is currently playing in and sets the score obtained in that level.
  + void setLevel();
    - It is a public member.
    - It updates the player’s current level.

## NON MEMBER FUNCTIONS:

* + void addToQueue(MONSTER &Node);
    - Adds a node to a queue of the type MONSTER, a self-referential class.
  + void deleteFromQueue();
    - Deletes a node from the front of the queue.
  + void RefreshQueue();
    - Refreshes the queue by resetting front and rear of the queue to 0;
  + void SetMaxScoreInLevel();
  + It sets a global variable, which keeps track of the maximum score, according to the level the player is in.
* char\* LevelName();
  + Returns a pointer to a string literal holding the name of the image file of the background based on the level the player is in.
* void MonsterDesign();
  + It determines the initial positions of the monsters based on the level the user is in.
* int distance(int x1, int y1, int x2, int y2);
  + It returns the distance between the points determined by the set of coordinates (x1,y1) and (x2,y2).
* void Update(PACMAN &P);
  + It updates the position of the Pacman by calling the function stepType1() (member function of the class PACMAN).
  + It traverses through the queue of MONSTER type objects and calls the function Step (member function of the class MONSTER) for each object.
  + It also checks for collision between Pacman and any one of the Monsters.
  + It updates and prints the score on the screen.
* void Background();
  + It changes and updates the background of the game window based on the level the player is in.
  + It changes background by reading and loading an image file.
* int main()
  + Creates a menu, giving the user a list of actions he can do.
  + Declares objects of PACMAN, MONSTER and PLAYER classes and utilizes them.
  + Reads from a dat file containing player details in order to log the user into his account.
  + Calls the above mentioned functions as required.
  + Updates the dat file containing player details with any progress the player has made.

# SOURCE CODE-

**Contents of “player.h” :-**

#ifndef PLAYER\_H

#define PLAYER\_H

extern int score , level;

extern const int noLevels;

using namespace std;

class PLAYER{

public:

char name[30];

char passcode[12];

int step;

int score[50]; //score of each level

int level; //current level user is in. Starts from 1.

PLAYER (){

name[0]=0;

level=1;

for(int i=0;i<50;i++)

score[i]=0;

}

void create(){

char temp[12];

level=1;

cout<<"NAME: ";

cin>>name;

A:

cout<<"Enter PASSCODE : ";

cin>>(passcode);

cout<<"Re-enter PASSCODE :";

cin>>(temp);

if(strcmp(temp , passcode)==0)

cout<<"Your account has been created successfully"<<endl;

else{

system("cls");

goto A;

}

}

void login(){

cout<<"Enter Your Name: ";

cin>>name;

cout<<"Enter Passcode: ";

cin>>passcode;

}

int TotalScore(){

int sum=0;

for(int i=0;i<50;i++)

sum+=score[i];

return sum;

}

void display(){

cout<<"NAME: "<<name<<endl<<endl;

cout<<"PASSCODE: "<<passcode<<endl<<endl;

if(level<=noLevels)

cout<<"LEVEL: "<<level<<endl<<endl;

else

cout<<"LEVEL: "<<"Finished All Levels!!!"<<endl<<endl;

for(int i=0;(i<level && i<noLevels);i++)

cout<<"SCORE - LEVEL "<<(i+1)<<": "<<score[i]<<endl<<endl;

cout<<"TOTAL SCORE: "<<TotalScore()<<endl<<endl;

}

void incrementLevel(){

level++;

}

void SetScore(int lev){

score[lev - 1]=::score;

}

void SetLevel(){

level=::level;

}

};

#endif

**END OF “player.h”**

**Contents of “object.h” :-**

#ifndef OBJECT\_H

#define OBJECT\_H

class OBJECT{

protected:

int area;

int cox,coy; //position coordinates

char key;

int step; //movement

void initialize();

public:

int getCOX(){

return cox;

}

int getCOY(){

return coy;

}

void Step();

};

#endif

**END OF “object.h”**

**Contents of “pacman.h” :-**

#ifndef PACMAN\_H

#define PACMAN\_H

extern int score , key;

class PACMAN : public OBJECT{

private:

void \*rightPac , \*leftPac, \*upPac, \*downPac, \*p[3];

int index ,keyBoardPressed;

char chk;

void initialize(){

cox=20;

coy=20;

index=0;

step=10;

keyBoardPressed=0;

chk=0;

//black

setcolor(BLACK);

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(685,200,15,15);

area = imagesize(0, 0, 40 , 40);

p[2] = malloc(area);

getimage(670 , 185 , 700 , 215 , p[2]);

putimage(670 , 185 , p[2] , XOR\_PUT);

//closed mouth

setcolor(BLACK);

setfillstyle(SOLID\_FILL,YELLOW);

fillellipse(685,200,15,15);

area = imagesize(0, 0, 40 , 40);

p[1] = malloc(area);

getimage(670 , 185 , 700 , 215 , p[1]);

putimage(670 , 185 , p[1] , XOR\_PUT);

//right

setcolor(BLACK);

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(685,200,15,15);

setfillstyle(SOLID\_FILL,YELLOW);

sector(685 , 200 , 30 , 330 , 15 , 15);

area = imagesize(0, 0, 40 , 40);

rightPac = malloc(area);

getimage(670 , 185 , 700 , 215 , rightPac);

putimage(670 , 185 , rightPac , XOR\_PUT);

//left

setcolor(BLACK);

setfillstyle(SOLID\_FILL,YELLOW);

fillellipse(685,200,15,15);

setfillstyle(SOLID\_FILL,BLACK);

sector(685 , 200 , 150 , 210 , 15 , 15);

area = imagesize(0, 0, 40 , 40);

leftPac = malloc(area);

getimage(670 , 185 , 700 , 215 , leftPac);

putimage(670 , 185 , leftPac , XOR\_PUT);

//up

setcolor(BLACK);

setfillstyle(SOLID\_FILL,YELLOW);

fillellipse(685,200,15,15);

setfillstyle(SOLID\_FILL,BLACK);

sector(685 , 200 , 60 , 120 , 15 , 15);

area = imagesize(0, 0, 40 , 40);

upPac = malloc(area);

getimage(670 , 185 , 700 , 215 , upPac);

putimage(670 , 185 , upPac , XOR\_PUT);

//down

setcolor(BLACK);

setfillstyle(SOLID\_FILL,YELLOW);

fillellipse(685,200,15,15);

setfillstyle(SOLID\_FILL,BLACK);

sector(685 , 200 , 240 , 300 , 15 , 15);

area = imagesize(0, 0, 40 , 40);

downPac = malloc(area);

getimage(670 , 185 , 700 , 215 , downPac);

putimage(670 , 185 , downPac , XOR\_PUT);

p[0]=rightPac;

}

public:

PACMAN(){

initialize();

putimage(cox , coy , p[0] , COPY\_PUT);

}

PACMAN(int x, int y){

initialize();

cox=x;

coy=y;

putimage(cox , coy , p[0] , COPY\_PUT);

}

~PACMAN(){

erasePacman();

}

void clearPacman(){

putimage(cox , coy , p[2] , COPY\_PUT);

}

void erasePacman(){

if(p[0]==rightPac){

int sAngle=30, eAngle=330;

while(sAngle!=eAngle){

setcolor(BLACK);

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(cox+15,coy+15,15,15);

setfillstyle(SOLID\_FILL,YELLOW);

sector(cox+15,coy+15 , sAngle , eAngle , 15 , 15);

sAngle+=10;

eAngle-=10;

if(sAngle>=360){

sAngle-=360;

}else if(sAngle<0){

sAngle+=360;

}

if(eAngle>=360){

eAngle-=360;

}else if(eAngle<0){

eAngle+=360;

}

delay(50);

}

putimage(cox , coy , p[2] , COPY\_PUT);

}else if(p[0]==leftPac){

int sAngle=150, eAngle=210;

while(sAngle!=eAngle){

setcolor(BLACK);

setfillstyle(SOLID\_FILL,YELLOW);

fillellipse(cox+15,coy+15,15,15);

setfillstyle(SOLID\_FILL,BLACK);

sector(cox+15,coy+15, sAngle , eAngle , 15 , 15);

sAngle-=10;

eAngle+=10;

if(sAngle>=360){

sAngle-=360;

}else if(sAngle<0){

sAngle+=360;

}

if(eAngle>=360){

eAngle-=360;

}else if(eAngle<0){

eAngle+=360;

}

delay(50);

}

putimage(cox , coy , p[2] , COPY\_PUT);

}else if(p[0]==upPac){

int sAngle=60, eAngle=120;

while(sAngle!=eAngle){

setcolor(BLACK);

setfillstyle(SOLID\_FILL,YELLOW);

fillellipse(cox+15,coy+15,15,15);

setfillstyle(SOLID\_FILL,BLACK);

sector(cox+15,coy+15, sAngle , eAngle , 15 , 15);

sAngle-=10;

eAngle+=10;

if(sAngle>=360){

sAngle-=360;

}else if(sAngle<0){

sAngle+=360;

}

if(eAngle>=360){

eAngle-=360;

}else if(eAngle<0){

eAngle+=360;

}

delay(50);

}

putimage(cox , coy , p[2] , COPY\_PUT);

}else if(p[0]==downPac){

int sAngle=240, eAngle=300;

while(sAngle!=eAngle){

setcolor(BLACK);

setfillstyle(SOLID\_FILL,YELLOW);

fillellipse(cox+15,coy+15,15,15);

setfillstyle(SOLID\_FILL,BLACK);

sector(cox+15,coy+15, sAngle , eAngle , 15 , 15);

sAngle-=10;

eAngle+=10;

if(sAngle>=360){

sAngle-=360;

}else if(sAngle<0){

sAngle+=360;

}

if(eAngle>=360){

eAngle-=360;

}else if(eAngle<0){

eAngle+=360;

}

delay(50);

}

putimage(cox , coy , p[2] , COPY\_PUT);

}

}

int obstacleUp(){

int colour=0;

if((colour=getpixel(cox+15,coy-7))!=BLACK ){ //collided up-centre

return colour;

}else if((colour=getpixel(cox,coy-7))!=BLACK ){

//collided up-left

return colour;

}else if((colour=getpixel(cox+30,coy-7))!=BLACK ){

//collided up-right

return colour;

}else

return 0; //0 = no collision

}

int obstacleDown(){

int colour=0;

if((colour=getpixel(cox+15,coy+35))!=BLACK ){

//collided down-centre

return colour;

}else if((colour=getpixel(cox,coy+35))!=BLACK ){ //collided down-left

return colour;

}else if((colour=getpixel(cox+30,coy+35))!=BLACK ){

//collided down-right

return colour;

}else

return 0; //0 = no collision

}

int obstacleLeft(){

int colour=0;

if((colour=getpixel(cox-5,coy+15))!=BLACK ) //collided left-centre

return colour;

else if((colour=getpixel(cox-5,coy))!=BLACK ) //collided left-up

return colour;

else if((colour=getpixel(cox-5,coy+30))!=BLACK )

//collided left-bottom

return colour;

else

return 0; //0 = no collision

}

int obstacleRight(){

int colour=0;

if((colour=getpixel(cox+37,coy+15))!=BLACK ) //collided right-centre

return colour;

else if((colour=getpixel(cox+37,coy))!=BLACK ) //collided right-up

return colour;

else if((colour=getpixel(cox+37,coy+30))!=BLACK )

//collided right-bottom

return colour;

else

return 0; //0 = no collision

}

void stepType1(){ //auto move.... step by 1 frame forward

{

while(kbhit()){

key=getch();

::key=key;

}

{

putimage(cox , coy , p[2] , COPY\_PUT); //erase old image

int colour =0; //store the colour of obstacle

/\*

colour:-

0=Black=Background=no obstacle

Green=Boundary

Red=Food

\*/

if((key=='w'||key=='W') && ((colour=obstacleUp())==0 || colour==RED)){

//up. i.e go up if no obstacle or obstacle is food

coy-=step;

p[0]=upPac;

if(colour==RED)

score++;

}else if((key=='s'||key=='S') && ((colour=obstacleDown())==0 || colour==RED)){ //down

coy +=step;

p[0]=downPac;

if(colour==RED)

score++;

}else if((key=='d'||key=='D') && ((colour=obstacleRight())==0 || colour==RED)){ //right

cox +=step;

p[0]=rightPac;

if(colour==RED)

score++;

}else if((key=='a'||key=='A') && ((colour=obstacleLeft())==0 || colour==RED)){ //left

cox -=step;

p[0]=leftPac;

if(colour==RED)

score++;

}else if(key==27){ //Esc(escape) key: Exit

//break;

::key=27;

}

putimage(cox , coy , p[index] , COPY\_PUT); //put new image

index=!index;

delay(100);

}

}

}

};

#endif

**END OF “pacman.h”**

**Contents of “monster.h” :-**

#ifndef MONSTER\_H

#define MONSTER\_H

class MONSTER : public OBJECT{

private:

void \*monster , \*m[1];

time\_t lastGenerated;

void initialize(){

NEXT=0; //self referential

cox=20;

coy=20;

step=10;

key=0;

lastGenerated=0;

//monster image

setcolor(WHITE);

setfillstyle(SOLID\_FILL,BLUE);

sector(685 , 200 , 0 , 180 , 15 , 15); //head

bar(670,200,700,230); //body

setcolor(BLUE);

line(671,199,699,199);

setfillstyle(SOLID\_FILL,WHITE);

fillellipse(680,200,5,10); //eye

fillellipse(690,200,5,10); //eye

setcolor(WHITE);

line(670,200,670,230);

line(700,200,700,230);

line(670,230,700,230);

area = imagesize(0, 0, 30 , 45);

monster = malloc(area);

getimage(670 , 185 , 700 , 230 , monster);

putimage(670 , 185 , monster , XOR\_PUT);

m[0]=monster;

}

public:

MONSTER \*NEXT; //self referential class, queue

MONSTER(){

initialize();

putimage(cox , coy , m[0] , COPY\_PUT);

}

MONSTER(int x, int y){

initialize();

cox=x;

coy=y;

putimage(cox , coy , m[0] , COPY\_PUT);

}

~MONSTER(){

eraseMonster();

}

void clearMonster(){

putimage(cox , coy , m[0] , XOR\_PUT);

}

void eraseMonster(){

clearMonster();

}

int obstacleUp(){

int colour=0;

if((colour=getpixel(cox+15,coy-7))!=BLACK ){ //collided up-centre

return colour;

}else if((colour=getpixel(cox,coy-7))!=BLACK ){ //collided up-left

return colour;

}else if((colour=getpixel(cox+30,coy-7))!=BLACK ){

//collided up-right

return colour;

}else

return 0; //0 = no collision

}

int obstacleDown(){

int colour=0;

if((colour=getpixel(cox+15,coy+55))!=BLACK ){

//collided down-centre

return colour;

}else if((colour=getpixel(cox,coy+55))!=BLACK ){ //collided down-left

return colour;

}else if((colour=getpixel(cox+30,coy+55))!=BLACK ){

//collided down-right

return colour;

}else

return 0; //0 = no collision

}

int obstacleLeft(){

int colour=0;

if((colour=getpixel(cox-5,coy+15))!=BLACK ) //collided left-centre

return colour;

else if((colour=getpixel(cox-5,coy))!=BLACK ) //collided left-up

return colour;

else if((colour=getpixel(cox-5,coy+45))!=BLACK )

//collided left-bottom

return colour;

else

return 0; //0 = no collision

}

int obstacleRight(){

int colour=0;

if((colour=getpixel(cox+37,coy+15))!=BLACK ) //collided right-centre

return colour;

else if((colour=getpixel(cox+37,coy))!=BLACK ) //collided right-up

return colour;

else if((colour=getpixel(cox+37,coy+45))!=BLACK )

//collided right-bottom

return colour;

else

return 0; //0 = no collision

}

void Step(){ //auto move.... step by 1 frame forward

{

if((clock()-lastGenerated)>1000){

//time gap between generating random numbers is 1000 ms

key = rand() % 4 + 1; //random number, 1 to 4...

lastGenerated=clock();

}

/\*

up = 1

down = 2

left = 3

right = 4

\*/

{

putimage(cox , coy , m[0] , XOR\_PUT); //erase old image

int colour=0;

if((key==1) && ((colour=obstacleUp())==0 || colour==RED)){

//up, if no obstacle or if obstacle is food

coy-=step;

}else if((key==2) && ((colour=obstacleDown())==0 || colour==RED)){ //down

coy +=step;

}else if((key==4) && ((colour=obstacleRight())==0 || colour==RED)){ //right

cox +=step;

}else if((key==3) && ((colour=obstacleLeft())==0 || colour==RED)){ //left

cox -=step;

}

putimage(cox , coy , m[0] , XOR\_PUT); //put new image

}

}

}

};

MONSTER \*Front=0,\*Rear=0;

void addToQueue(MONSTER &Node){

if(Front==0){

Front = Rear = &Node;

Rear->NEXT=0;

}else{

Rear->NEXT= &Node;

Rear = Rear->NEXT;

Rear->NEXT=0;

}

}

void deleteFromQueue(){

if(Front==0){

return;

}else if(Front==Rear){

MONSTER \*t=Front;

Rear = Front =0;

delete t;

}else{

MONSTER \*t=Front;

Front = Front->NEXT; t->NEXT=0;

delete t;

}

}

void RefreshQueue(){

Front=Rear=0;

}

#endif

**END OF “monster.h”**

**Contents of the CPP file:-**

#include<iostream>

#include<string.h>

#include<graphics.h>

#include<conio.h>

#include<fstream>

#include<stdlib.h>

#include<time.h>

#include<math.h>

#include "object.h"

#include "pacman.h"

#include "monster.h"

#include "player.h"

using namespace std;

const int maxNoPlayers = 70;

const int noLevels = 5;

int key,score=0,level=1;

void \*background=0;

//LEVELS

int maxScoreInLevel=0;

void SetMaxScoreInLevel(){

if (level==1){

maxScoreInLevel=180;

}else if (level==2){

maxScoreInLevel=180;

}else if (level==3){

maxScoreInLevel=180;

}else if (level==4){

maxScoreInLevel=270;

}else if (level==5){

maxScoreInLevel=270;

}

}

char\* LevelName(){

if (level==1){

return "level1.bmp";

}else if (level==2){

return "level2.bmp";

}else if (level==3){

return "level3.bmp";

}else if (level==4){

return "level4.bmp";

}else if (level==5){

return "level5.bmp";

}

}

//

//MONSTER design for levels

void MonsterDesign(){

if (level==1){

RefreshQueue();

static MONSTER M1(300,120), M2(1200,120), M3(1200,500),

M4(200,580), M5(640,270), M6(900,650);

//Form a Queue of Monsters

addToQueue(M1);

addToQueue(M2);

addToQueue(M3);

addToQueue(M4);

addToQueue(M5);

addToQueue(M6);

//

}else if (level==2){

RefreshQueue();

static MONSTER M1(300,120), M2(1050,140), M3(750,600),

M4(200,580), M5(600,600), M6(600,100);

//Form a Queue of Monsters

addToQueue(M1);

addToQueue(M2);

addToQueue(M3);

addToQueue(M4);

addToQueue(M5);

addToQueue(M6);

//

}else if (level==3){

RefreshQueue();

static MONSTER M1(300,600), M2(450,150), M3(1000,600),

M4(1000,100),M5(460,450);

//Form a Queue of Monsters

addToQueue(M1);

addToQueue(M2);

addToQueue(M3);

addToQueue(M4);

addToQueue(M5);

//

}else if (level==4){

RefreshQueue();

static MONSTER M1(300,120), M2(1200,120), M3(1000,500),

M4(200,600), M5(650,300);

//Form a Queue of Monsters

addToQueue(M1);

addToQueue(M2);

addToQueue(M3);

addToQueue(M4);

addToQueue(M5);

//

}else if (level==5){

RefreshQueue();

static MONSTER M1(200,160), M2(330,400), M3(700,500),

M4(1000,280), M5(1200,500), M6(400,630);

//Form a Queue of Monsters

addToQueue(M1);

addToQueue(M2);

addToQueue(M3);

addToQueue(M4);

addToQueue(M5);

addToQueue(M6);

//

}

}

//

PLAYER Player1;

int distance(int x1, int y1, int x2, int y2){ //distance between 2 points

return int(sqrtl(((x1-x2)\*(x1-x2))+(y1-y2)\*(y1-y2)));

}

void Update(PACMAN &P){

P.stepType1();

MONSTER \*M=Front;

while(M!=0){

if(distance(P.getCOX(),P.getCOY(),M->getCOX(),M->getCOY())<600){

//distance between pacman and monster is less than 600

M->Step();

if(M->obstacleUp()==YELLOW||M->obstacleDown()==YELLOW||

M->obstacleLeft()==YELLOW||M->obstacleRight()==YELLOW){

//PACMAN touching monster

key=27;

return;

}

}

M=M->NEXT;

}

char pts[30];

sprintf (pts, "Score: %d", score); //print score

outtextxy(30,700,pts);

}

void Background(){

int area=0;

readimagefile(LevelName(),0,0,getmaxx(),getmaxy());

//load required level on screen

area = imagesize(0, 0, getmaxx() , getmaxy());

background = malloc(area);

getimage(0 , 0 , getmaxx() , getmaxy(), background);

}

void gotoxy( int column, int line )

{

COORD coord;

coord.X = column;

coord.Y = line;

SetConsoleCursorPosition(

GetStdHandle( STD\_OUTPUT\_HANDLE ),

coord

);

}

int main()

{

char choice;

int chk = 0,window;

fstream f;

cout<<"MENU:"<<endl;

cout<<"1 : CREATE ACCOUNT "<<endl;

cout<<"2 : LOGIN"<<endl;

cout<<"3 : DELETE ACCOUNT"<<endl;

cout<<"4 : VIEW ACCOUNT DETAILS"<<endl;

cout<<"5 : SCORES"<<endl;

cout<<"6 : INSTRUCTIONS"<<endl;

cout<<"7 : EXIT"<<endl<<endl;

cout<<"Enter your choice: ";

cin>>choice;

if(choice=='1') //create account //Appending to file

{

f.open("Accounts.dat" , ios::out|ios::binary|ios::app);

system("cls");

Player1.create();

f.write((char\*)&Player1 , sizeof(Player1));

f.close();

cout<<"Account Created Successfully!!"<<endl<<endl<<"Starting Game

in 2 seconds...";

delay(2000);

goto Z;

}

if(choice=='2') //login i.e. Reading From File

{

int chk = 0,i=0,j=0,N=0,x=12,y=0;

char ch,name[50]={0},passcode[50]={0},k;

PLAYER temp, List[maxNoPlayers];

f.open("Accounts.dat" , ios::in|ios::binary);

//Creating and displaying Suggestions List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

i=0;

while(f.read((char\*)&List[i] , sizeof(List[i])))

{

i++;

N++;

}

i=0;

A: system("cls");

cout<<"Enter Name: "<<name;

gotoxy(50,0);

cout<<"Suggestions:";

while(true){

gotoxy(x,0);

k=getche();

if(k=='\r')

break;

else if(k=='\b'){

if(i!=0){

i--;

x--;

}

name[i]=0;

goto A;

}

name[i++]=k;

x++;

name[i]=0;

//clear previous suggestions

j=0;

y=0;

while(j!=N){

y+=2;

gotoxy(50,y);

cout<<" ";

j++;

}

//output suggestions

j=0;

y=0;

while(j!=N){ //output suggestions

if(strstr(List[j].name , name)!=0){

y+=2;

gotoxy(50,y);

cout<<List[j].name;

}

j++;

}

}

i=0;

L: gotoxy(0,2);

cout<<"Enter Passcode: "<<passcode;

while(true){

k=getche();

if(k=='\r')

break;

else if(k=='\b'){

if(i!=0)

i--;

passcode[i]=0;

gotoxy(0,2);

cout<<"Enter Passcode: ";

goto L;

}

passcode[i++]=k;

passcode[i]=0;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

j=0;

while(j!=N) //Matching Data for login

{

if(strcmp(passcode,List[j].passcode)==0 &&

strcmp(name,List[j].name)==0)

{

Player1 = List[j];

level=Player1.level;

chk++;

break;

}

j++;

}

f.close();

if(chk==0)

{

i=name[0]=passcode[0]=0;

x=12;

cout<<endl<<endl<<"Login Unsuccessful!!"<<endl;

cout<<"Do You want to RETRY? (Y/N)"<<endl;

cin>>ch;

if(ch=='y'||ch=='Y')

{

goto A;

}

if(ch=='n' ||'N')

{

exit(0);

}

}

if(chk!=0)

{

if(Player1.level>noLevels){ //if player has finished all levels

cout<<endl<<endl<<"Login Successful!!";

cout<<endl<<"You have FINISHED all levels.";

cout<<endl<<"If you continue, you will be loaded to LEVEL 1.";

cout<<endl<<"Your scores will be Reset.";

cout<<endl<<"CONTINUE? (Y/N): ";

cin>>ch;

if(ch=='y'||ch=='Y')

{

level=Player1.level=1;

for(int k=0;k<noLevels;k++){

Player1.score[k]=0;

}

cout<<endl<<endl<<"Starting Game in 2 seconds...";

delay(2000);

goto Z;

}

if(ch=='n' ||'N')

{

exit(0);

}

}else{

cout<<endl<<endl<<"Login Successful!!"<<endl<<endl<<"Starting

Game in 2 seconds...";

delay(2000);

goto Z;

}

}

}

if(choice=='3') //delete

{

int chk = 0,i=0,j=0,N=0,x=12,y=0,pos=-1;

char ch,name[50]={0},passcode[50]={0},k;

PLAYER temp, List[maxNoPlayers];

f.open("Accounts.dat" , ios::in|ios::binary);

//Creating and displaying Suggestions List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

i=0;

while(f.read((char\*)&List[i] , sizeof(List[i])))

{

i++;

N++;

}

i=0;

A2: system("cls");

cout<<"Enter Name: "<<name;

gotoxy(50,0);

cout<<"Suggestions:";

while(true){

gotoxy(x,0);

k=getche();

if(k=='\r')

break;

else if(k=='\b'){

if(i!=0){

i--;

x--;

}

name[i]=0;

goto A2;

}

name[i++]=k;

x++;

name[i]=0;

//clear previous suggestions

j=0;

y=0;

while(j!=N){

y+=2;

gotoxy(50,y);

cout<<" ";

j++;

}

//output suggestions

j=0;

y=0;

while(j!=N){ //output suggestions

if(strstr(List[j].name , name)!=0){

y+=2;

gotoxy(50,y);

cout<<List[j].name;

}

j++;

}

}

i=0;

L2: gotoxy(0,2);

cout<<"Enter Passcode: "<<passcode;

while(true){

k=getche();

if(k=='\r')

break;

else if(k=='\b'){

if(i!=0)

i--;

passcode[i]=0;

gotoxy(0,2);

cout<<"Enter Passcode: ";

goto L2;

}

passcode[i++]=k;

passcode[i]=0;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

j=0;

while(j!=N) //Matching Data for login

{

if(strcmp(passcode,List[j].passcode)==0 &&

strcmp(name,List[j].name)==0)

{

Player1 = List[j];

pos=j;

chk++;

break;

}

j++;

}

f.close();

if(chk==0)

{

i=name[0]=passcode[0]=0;

x=12;

cout<<endl<<endl<<"Login Unsuccessful!!"<<endl;

cout<<"Do You want to RETRY? (Y/N)"<<endl;

cin>>ch;

if(ch=='y'||ch=='Y')

{

goto A2;

}

if(ch=='n' ||'N')

{

exit(0);

}

}

if(chk!=0)

{

cout<<endl <<"Logged In!!"<<endl<<endl<<"DELETE? (Y/N) :";

cin>>ch;

if(ch=='y'||ch=='Y')

{

f.close();

f.open("Accounts.dat",ios::in|ios::binary);

fstream tempFile ("temp.dat" , ios::out|ios::binary);

while(f.read((char\*)&temp , sizeof(temp)))

{

if(strcmp(Player1.name, temp.name)==0 &&

strcmp(Player1.passcode, temp.passcode)==0){

}else{

tempFile.write((char\*)&temp , sizeof(temp));

}

}

f.close();

tempFile.close();

remove("Accounts.dat");

rename("temp.dat","Accounts.dat");

}

if(ch=='n' ||'N')

{

exit(0);

}

cout<<"ACCOUNT DELETED...";

delay(2000);

exit(0);

}

}

if(choice=='4') //account details

{

int chk = 0,i=0,j=0,N=0,x=12,y=0,pos=-1;

char ch,name[50]={0},passcode[50]={0},k;

PLAYER temp, List[maxNoPlayers];

f.open("Accounts.dat" , ios::in|ios::binary);

//Creating and displaying Suggestions List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

i=0;

while(f.read((char\*)&List[i] , sizeof(List[i])))

{

i++;

N++;

}

i=0;

A3: system("cls");

cout<<"Enter Name: "<<name;

gotoxy(50,0);

cout<<"Suggestions:";

while(true){

gotoxy(x,0);

k=getche();

if(k=='\r')

break;

else if(k=='\b'){

if(i!=0){

i--;

x--;

}

name[i]=0;

goto A3;

}

name[i++]=k;

x++;

name[i]=0;

//clear previous suggestions

j=0;

y=0;

while(j!=N){

y+=2;

gotoxy(50,y);

cout<<" ";

j++;

}

//output suggestions

j=0;

y=0;

while(j!=N){ //output suggestions

if(strstr(List[j].name , name)!=0){

y+=2;

gotoxy(50,y);

cout<<List[j].name;

}

j++;

}

}

i=0;

L3: gotoxy(0,2);

cout<<"Enter Passcode: "<<passcode;

while(true){

k=getche();

if(k=='\r')

break;

else if(k=='\b'){

if(i!=0)

i--;

passcode[i]=0;

gotoxy(0,2);

cout<<"Enter Passcode: ";

goto L3;

}

passcode[i++]=k;

passcode[i]=0;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

j=0;

while(j!=N) //Matching Data for login

{

if(strcmp(passcode,List[j].passcode)==0 &&

strcmp(name,List[j].name)==0)

{

Player1 = List[j];

pos=j;

chk++;

break;

}

j++;

}

f.close();

if(chk==0)

{

i=name[0]=passcode[0]=0;

x=12;

cout<<endl<<endl<<"Login Unsuccessful!!"<<endl;

cout<<"Do You want to RETRY? (Y/N)"<<endl;

cin>>ch;

if(ch=='y'||ch=='Y')

{

system("cls");

goto A3;

}

if(ch=='n' ||'N')

{

exit(0);

}

}

if(chk!=0)

{

cout<<endl<<endl<<"Logged In!!"<<endl<<endl<<"Displaying...";

delay(1500);

system("cls");

Player1.display();

getche();

exit(0);

}

}

if(choice=='5') //scores

{

f.close();

f.open("Accounts.dat" , ios::in|ios::binary);

int i=0,j=0,N=0;

PLAYER temp, List[maxNoPlayers];

i=0;

while(f.read((char\*)&List[i] , sizeof(List[i])))

{

i++;

N++;

}

f.close();

for(i=0;i<N;i++){ //Sort- descending

for(j=i;j<N;j++){

if(List[i].TotalScore() < List[j].TotalScore()){

temp = List[i];

List[i]=List[j];

List[j]=temp;

}

}

}

system("cls");

gotoxy(0,1);

cout<<"| NAME";

gotoxy(20,1);

cout<<"| SCORE";

gotoxy(30,1);

cout<<"| LEVEL |";

gotoxy(0,2);

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

gotoxy(0,0);

cout<<"|";

gotoxy(20,0);

cout<<"|";

gotoxy(30,0);

cout<<"|";

gotoxy(41,0);

cout<<"|";

gotoxy(0,2);

cout<<"|";

gotoxy(20,2);

cout<<"|";

gotoxy(30,2);

cout<<"|";

gotoxy(41,2);

cout<<"|";

gotoxy(0,3);

cout<<"|";

gotoxy(20,3);

cout<<"|";

gotoxy(30,3);

cout<<"|";

gotoxy(41,3);

cout<<"|";

for(i=0;i<N;i++){

gotoxy(0,i+4);

cout<<"| "<<List[i].name;

gotoxy(20,i+4);

cout<<"| "<<List[i].TotalScore();

gotoxy(30,i+4);

if(List[i].level<=noLevels)

cout<<"| "<<List[i].level<<" |";

else

cout<<"| "<<"FINISHED"<<" |";

}

gotoxy(0,i+4);

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

gotoxy(0,i+4);

cout<<"|";

gotoxy(20,i+4);

cout<<"|";

gotoxy(30,i+4);

cout<<"|";

gotoxy(41,i+4);

cout<<"|";

cout<<endl<<endl;

getche();

exit(0);

}

if(choice=='6') //instructions

{

system("cls");

fstream t("instructions.txt", ios::in);

char s[100];

while(t.getline(s,100,'\n')){

cout<<s<<endl;

}

t.close();

getche();

exit(0);

}

if(choice!='1' && choice!='2' && choice!='3' && choice!='4' && choice!='5'

&& choice!='6' && choice!='D')

choice='7';

if(choice=='7')

{

exit(0);

}

if(choice=='D') //developer test

{

cout<<endl<<".DEVELOPER TEST MODE."<<endl;

window=initwindow(getmaxwidth() , getmaxheight() , "Game:

PACMAN");

Lab: cleardevice();

Background(); //initialization for the background

cleardevice();

PACMAN P(699,48);

MonsterDesign();

srand(time(NULL));

//Printing Background

putimage(0,0,background, 2);

SetMaxScoreInLevel();

while(key!=27){

//till escape key is not pressed, scene keeps updating...

if(key ==51){ //'3' is pressed

score=maxScoreInLevel;

key=0;

}

Update(P);

if(score>=maxScoreInLevel){

//if player scores enough to advance to next level

Player1.SetScore(level);

if(level<noLevels){

//if current level is lesser than no. of levels, increment level.

level++;

SetMaxScoreInLevel();

score=0;

Player1.SetLevel();

//closegraph();

cleardevice();

outtextxy(400,300,"CONGRATULATIONS!! YOU HAVE

CLEARED THIS LEVEL. LOADING NEXT LEVEL...");

cout<<"CONGRATULATIONS!! YOU HAVE CLEARED THIS

LEVEL."<<endl<<"LOADING NEXT LEVEL...";

delay(2000);

goto Lable;

}else{

//closegraph();

system("cls");

cleardevice();

outtextxy(450,300,"CONGRATULATIONS!! YOU HAVE

FINISHED THE GAME. THANK YOU...");

delay(3000);

key=27;

}

}

}

}

exit(0);

Z: window=initwindow(getmaxwidth() , getmaxheight() , "Game: PACMAN");

N: cleardevice();

Background(); //initialization for the background

cleardevice();

PACMAN P(699,48);

MonsterDesign();

srand(time(NULL));

//Printing Background

putimage(0,0,background, 2);

SetMaxScoreInLevel();

while(key!=27){ //till escape key is not pressed, scene keeps updating...

Update(P);

if(score>=maxScoreInLevel){

//if player scores enough to advance to next level

Player1.SetScore(level);

if(level<noLevels){

//if current level is lesser than no. of levels, increment level.

level++;

SetMaxScoreInLevel();

score=0;

Player1.SetLevel();

cleardevice();

outtextxy(400,300,"CONGRATULATIONS!! YOU HAVE

CLEARED THIS LEVEL. LOADING NEXT LEVEL...");

cout<<"CONGRATULATIONS!! YOU HAVE CLEARED THIS

LEVEL."<<endl<<"LOADING NEXT LEVEL...";

delay(2000);

goto N;

}else{

//closegraph();

system("cls");

cleardevice();

outtextxy(450,300,"CONGRATULATIONS!! YOU HAVE

FINISHED THE GAME. THANK YOU...");

delay(3000);

key=27;

}

}

}

//Modify Player Data in File...

//UPDATING file

Player1.SetLevel();

Player1.SetScore(level);

if(Player1.level==noLevels && score>=maxScoreInLevel){

/\*

if player has finished last level then player's current level is set to

maximum level +1 to indicate the same

\*/

level=noLevels+1;

Player1.SetLevel();

score=0;

Player1.SetScore(level);

}

f.open("Accounts.dat" , ios::out|ios::in|ios::binary);

PLAYER temp;

int siz = sizeof(PLAYER)\*-1;

while(f.read((char\*)&temp , sizeof(temp)))

{

if(strcmp(Player1.name,temp.name)==0) //Matching Player

{

f.seekp(siz, ios::cur);

f.write((char\*)&Player1 , sizeof(Player1));

break;

}

}

f.close();

}

**END OF CPP file**