**File: Include.h**

#ifndef INCLUDE\_H

#define INCLUDE\_H

#include <fstream.h>

#include <conio.h>

#include <string.h>

#include <stdio.h>

#include <graphics.h>

#include <math.h>

#include <ctype.h>

#include "palash\include\keycode.h"

#include "palash\include\strfun.h"

#include "palash\include\button.h"

#include "palash\include\label.h"

#include "palash\include\exp.h"

#include "palash\include\textbox.h"

#include "palash\include\image.h"

#include "palash\include\sort.h"

#include "palash\include\shm.h"

#include "palash\include\field.h"

#include "palash\include\pong.h"

#include "palash\include\snake.h"

#include "palash\include\graph.h"

#include "palash\include\tic.h"

#endif

**File: KeyCode.h**

#ifndef KEYCODE\_H

#define KEYCODE\_H

#define K\_UP 72

#define K\_DOWN 80

#define K\_LEFT 75

#define K\_RIGHT 77

#define K\_ENTER 13

#define K\_ESC 27

#endif

**File: Main.CPP**

#include "palash/include.h"

void main(){

iGraph(); cleardevice();settextstyle(4,0,5); setcolor(RED); settextjustify(1,1);

outtextxy(320,25,"Welcome to THE Project By"); drawImage("palash/profile.img",170,77);

settextstyle(10,0,3); setcolor(LIGHTBLUE); delay(10);

outtextxy(320,420,"Made Under Guidance of"); drawImage("palash/teacher.img",550,350);

clrKeyBuff(); getch();mchar flag=1,flag2;

while(flag)

{ flag2=1; cleardevice(); clrKeyBuff(); Button button[10];

button[0].initialize(25, 15, 205, 155,0,0,"Graphs"); button[1].initialize(230, 15, 410, 155,0,0,"Snakes");

button[2].initialize(435, 15, 615, 155,0,0,"Pong"); button[3].initialize(25, 170, 205, 310,0,0,"Calculator");

button[5].initialize(435, 170, 615, 310,0,0,"Sorting"); button[6].initialize(25, 325, 205, 465,0,0,"SHM");

button[7].initialize(230, 325, 410, 465,0,0,"Feild Lines");

button[8].initialize(435, 325, 615, 465,0,0,"Tic Tac Toe");

for(int i = 0; i<9;i++){

button[i].BorderColor[NORMAL] = BLUE-(i==4); button[i].BorderColor[FOCUS] = RED;

button[i].FillType[NORMAL]=button[i].FillType[FOCUS]=1;

button[i].TextColor[NORMAL]=button[i].TextColor[FOCUS]=RED;

button[i].Font[NORMAL]=button[i].Font[FOCUS]=6;

button[i].FontSize[NORMAL]=button[i].FontSize[FOCUS]=4; if(i!=4)button[i].make();

}

settextjustify(1,1); settextstyle(4,0,7); setcolor(LIGHTRED); outtextxy(320,190,"Welcome");

settextstyle(3,0,3); outtextxy(320,240,"Choose One"); settextstyle(0,0,1); setcolor(BLUE);

outtextxy(320,280,"(ESC to Quit)"); int curr =1; button[curr].gotFocus();

while(flag&&flag2){ int k = getch();

switch(k){

case K\_LEFT: button[curr].lostFocus();

if(--curr==-1)curr=8; if(curr==4)curr=3;

button[curr].gotFocus(); break;

case K\_RIGHT: button[curr].lostFocus();

if(++curr==9)curr=0; if(curr==4)curr=5;

button[curr].gotFocus(); break;

case K\_DOWN: button[curr].lostFocus();

curr+=3; if(curr>8)curr-=9; if(curr==4)curr=7;

button[curr].gotFocus(); break;

case K\_UP: button[curr].lostFocus();

curr-=3; if(curr<0)curr+=9; if(curr==4)curr=1;

button[curr].gotFocus(); break;

case K\_ENTER: switch(curr){

case 0: playGraph(); break;

case 1: playSnake(); break;

case 2: playPong("Player"); break;

case 3: playCalc(); break;

case 5: playSort(); break;

case 6: playSHM(); break;

case 7: playField(); break;

case 8: playTicTacToe(); break;

}; flag2=0; break;

case K\_ESC: flag=0; break;

}; } }

cleardevice(); closegraph();

}

**File: StrFun.h**

#ifndef STRFUN\_H

#define STRFUN\_H

char b\_w(double w, double min, double max){ return w>min && w<max; }

void iGraph(){ int gd = DETECT, gm; initgraph(&gd,&gm,"C:\\TC\\BGI"); }

void clrKeyBuff(){ while(kbhit()) getch(); }

const char FALSE = 0, false = 0, False = 0, TRUE = 1, true = 1, True = 1;

inline long ftoi(double a){ return fmod(a,1)<0.5?(a):(a+1); }

inline char\* ctos(char ch){ char \*str = new char[2]; sprintf(str, "%c", ch); return str; }

inline char\* itos(int ch){ char \*str = new char[20]; sprintf(str, "%i", ch); return str; }

inline char\* ftos(double ch){ char \*str = new char[20]; sprintf(str, "%lf", ch); return str; }

int ctoi(char ch){ return ch-48; }

long stoi(char \*ch){

long n = 0,j = 1, i; for(i = strlen(ch) - 1 ; i > 0 ; i--){ n += ctoi(ch[i]) \* j; j \*= 10; }

if(ch[0] != '-' && ch[0]!='+') n += ctoi(ch[0]) \* j; else if(ch[0] == '-') n \*= -1; return n;

}

double stof(char \* ch){

if(!strcmp(ch,"pi")||!strcmp(ch,"PI")) return M\_PI;

long p =-1,len=strlen(ch); double n =0;

for(long i = 0;i< len;i++)

if(ch[i]=='.'){ p=i; break; }

if(p==-1){strcat(ch,".0"); p=len++;len++; }

else if(p==0){ ch = rstrcat("0",ch); len++;p++; }

n += stoi(substr(ch, (ch[0]=='-'?1:0), p-1));

n += (n>=0?1:-1)\*( stoi(substr(ch,p+1,len-1))/pow(10,len-1-p));

n \*= (ch[0]=='-'?-1:1); return n;

}

char\* rstrcat(char \* str1, char\* str2){ char str3[100]; strcpy(str3,str1); strcat(str3,str2); return str3; }

char \* substr(char\* a,int start, int end){

char n[500]; for(int i=start;i<=end;i++) n[i-start]=a[i]; n[end-start+1]='\0'; return n;

}

void replace(char str[], char mat[], char rep[]){

int len = strlen(str),mlen = strlen(mat),rlen = strlen(rep);

for(int i = 0; i<len;i++) if(str[i]==mat[0]) if(!strcmp(substr(str,i,i+mlen-1),mat)) {

char buff[200] = "\0"; strcpy(buff,substr(str,i+mlen,len-1));

str[i] = '\0'; strcat(str,rep); strcat(str,buff); len+=rlen; i+=rlen-1;

} }

#endif

**File: Button.h**

#ifndef BUTTON\_H

#define BUTTON\_H

const char NORMAL = 0, FOCUS = 1; struct Coord { int X; int Y; };

double mean(double x, double y){ return (x + y) / 2; }

class Button

{

public: Coord Start, End; char \* Caption, Focus;

int FillType[2], BorderType[2], TextColor[2], FillColor[2], TextAlign[2], ForeColor[2], BackColor[2], BorderColor[2], BorderSize[2], FontSize[2], Font[2], BackgroundColor;

void initialize(int StartX, int StartY, int EndX, int EndY, int BColor = 0 , char Make = 0 , char \* Caption = "") {

Start.X = StartX; Start.Y = StartY; End.X = EndX; End.Y = EndY;

this->Caption = Caption; BackgroundColor = BColor;

TextColor[NORMAL] = 15; TextColor[FOCUS] = 15; FillColor[NORMAL] = 0; FillColor[FOCUS] = 0;

BorderColor[NORMAL] = 15; BorderColor[FOCUS] = 15; BorderSize[NORMAL] = 1; BorderSize[FOCUS] = 3;

FontSize[NORMAL] = 1; FontSize[FOCUS] = 1; TextAlign[NORMAL] = 5; TextAlign[FOCUS] = 5;

FillType[NORMAL] = 1; FillType[FOCUS] = 1; BorderType[NORMAL] = 0; BorderType[FOCUS] = 0;

Font[NORMAL] = 0; Font[FOCUS] = 0; Focus = FALSE; setVariables(); if(Make) make();

}

void make(){

erase(); setlinestyle(BorderType[Focus] , 0 , BorderSize[Focus]); setcolor(BorderColor[Focus]);

setfillstyle(FillType[Focus] , FillColor[Focus]); bar(Start.X , Start.Y , End.X , End.Y);

if(BorderSize[Focus])rectangle(Start.X , Start.Y , End.X , End.Y); fillPattern();

}

void fillPattern(){

int xp = (TextAlign[Focus]-1) % 3, yp = (TextAlign[Focus]-1) / 3;

int x = Start.X + xp\*(End.X - Start.X)/2, y = Start.Y + yp\*(End.Y - Start.Y)/2;

settextstyle(Font[Focus],0,FontSize[Focus]); yp=(yp-1)\*-1+1;

settextjustify(xp,yp); moveto(x,y+1); setcolor(TextColor[Focus]); outtext(Caption);

}

void erase(){

setfillstyle(SOLID\_FILL, BackgroundColor); setcolor(BackgroundColor);

setlinestyle(SOLID\_LINE, 0 , BorderSize[!Focus]); rectangle(Start.X, Start.Y, End.X, End.Y);

bar3d(Start.X, Start.Y, End.X, End.Y,0,0);

}

void gotFocus(){ Focus = TRUE; make(); }

void lostFocus(){ Focus = FALSE; make(); }

};

#endif

**File: TextBox.h**

#ifndef TEXTBOX\_H

#define TEXTBOX\_H

class TextBox

{

public: int len, sx , sy, w , h, key, bcolor, fcolor, i, endtype; char text[100], ch, flag, retend, retbksp;

void initialize(int ux=1,int uy=1,int l=50,char t[] = "",int fc=15, int bc=0){

len = l; key = -1; retbksp=0; sx = ux; sy = uy; flag = 1; i = 0; retend = 0; strcpy(text,t); bcolor = bc;

endtype = 0; fcolor = fc; w = textwidth(" "); h = textheight(" "); outtextxy(sx,sy,text);

}

char \* focus(){

moveto(sx + i\*w, sy); setcolor(fcolor); endtype = 0;

while(flag){

if(i==len&&retend){ endtype=1;break; } ch = getch();

switch(ch){

case 0: key=1;break;

case '\b': if(i != 0){ moveto(getx() - w, gety()); setcolor(bcolor);

outtext(ctos(text[i-1])); setcolor(fcolor);

moveto(getx() - w, gety()); text[--i]='\0'; }

else if(retbksp) endtype = flag--; break;

case '\n': case '\r': endtype = 0; flag = 0; break;

default: if(i<len&&key) {

text[i]=ch; text[i + 1]='\0'; outtextxy(sx, sy, text); moveto(sx + (w \* (++i)), sy); }

};

};

if(key>=0) key--; flag = 1; return text;

}

void setText(char t[]){

setcolor(bcolor); outtextxy(sx,sy,text); strcpy(text, t); setcolor(fcolor); outtextxy(sx,sy,text);i=strlen(text);

}

};

#endif

**File: Label.h**

#ifndef LABEL\_H

#define LABEL\_H

class Label{ char text[100];

public: int x, y, font, color, fsize, bcolor, justx, justy, dir;

Label(int x, int y, char t[] = " ", int f = 0,int fs =1,int fc =15,int bc=0,int jx = LEFT\_TEXT, int jy = TOP\_TEXT, int d=0)

{ this->x=x;this->y=y;font=f;strcpy(text,t);color=fc;fsize=fs;bcolor=bc;justx=jx;justy=jy;dir=d;

this->setText(t); }

void setText(char \* t) {

settextjustify(justx,justy); settextstyle(font,dir,fsize); setcolor(bcolor);

outtextxy(x,y,text); strcpy(text,t); setcolor(color); outtextxy(x,y,text);

}

char \* getText() { char t[100]; strcpy(t,text); return t; }

void changePos(int x, int y) {

char \* t = this->getText(); this->setText(""); this->x = x; this->y = y; setText(t); }

int width(){ return textwidth(text); }

int height(){ return textheight(text); }

};

#endif

**File: Image.h**

#ifndef IMAGE\_H

#define IMAGE\_H

void drawImage(char name[], int x=0, int y=0){

ifstream a(name,ios::binary); int i=0,j=0,no,k,l; char n[2],c;

while(!a.eof()) { no=1;int f=0;

while(a.read((char\*)&n,1)){

if(n[0]>=48&&n[0]<=57) { if(!f++) no=0; no=no\*10+n[0]-48; }

else break;

} c=n[0];

for(k = 0 ; k <= 16 ; k++){

if(k==16){ i++; j=0; }

else if(c==(char(65+k))) { for(l=0;l<no;l++) putpixel(i+x, j+++y, k); break; }

} } }

#endif

**File: Exp.h**

#ifndef EXP\_H

#define EXP\_H

double exp(char e[]){ return solve(e); }

char equals(char a,char b[]){

int f=0; for(int i = 0;i<strlen(b);i++) if(b[i]==a){f=1;break;} return f;

}

long nt(char b[],int s){

int l=0,len=strlen(b);

if(b[s]=='(') l=1;

for(long i = s+1;i<len;i++){

if(b[i]=='(') l++; else if(b[i]==')'){ l--; if(l<0) break; }

else if((b[i]=='+'||b[i]=='-')&&l==0) if(isdigit(b[i-1])||b[i-1]=='('||b[i-1]==')') break;

} return i;

}

double solve(char e[]){

double sum=0; long next,len=strlen(e),i=0;char sgn; char sgns[]="+-/^\*%";

if(e[0]=='('){ int l=1;

for(i = 1; i<len;i++) if(e[i]=='(') l++; else if(e[i]==')') if(!--l) break; sum = exp(substr(e,1,i-1)); }

else if(e[0]=='s') {

if(!strcmp("sin",substr(e,0,2))){

next = nt(e,3); double s =exp(substr(e,3,next-1)); sum = sin(s); i=next; }

} else if(e[0]=='c') {

if(!strcmp("cos",substr(e,0,2))) {

next = nt(e,3); double s =exp(substr(e,3,next-1)); sum = cos(s); i=next; }

}

else if(e[0]=='t') {

if(!strcmp("tan",substr(e,0,2))){

next = nt(e,3); double s =exp(substr(e,3,next-1)); sum = tan(s); i=next; }

}

else{

for(i=1;i<len;i++) if(equals(e[i],sgns)) break;

sum = stof(substr(e,0,i-1)); i=1; }

for(;i<len;i++) {

sgn=e[i];

if(sgn=='+'){ sum+=exp(substr(e,i+1,len-1));break; }

else if(sgn=='-'){ sum+=exp(substr(e,i,len-1)); break; }

else if(sgn=='\*') { next = nt(e,i+1); sum\*=exp(substr(e,i+1,next-1)); i=next-1; }

else if(sgn=='/') {

next=nt(e,i+1); double d=exp(substr(e,i+1,next-1));

if(d)sum/=d; else return 0; i=next-1;

}

else if(sgn=='^'){ next = nt(e,i+1); sum = pow(sum,exp(substr(e,i+1,next-1))); i=next-1; }

} return sum;

}

void playCalc(){

cleardevice(); char e[100]; gotoxy(10,12); cout<<"Enter Expression: "; cin>>e;

gotoxy(10,13); cout<<"The Solution is "<<solve(e); getch();

}

#endif

**File: Sort.h**

#ifndef SORT\_H

#define SORT\_H

void draw(int a[]){ int w= 60, mf=50;

setfillstyle(1,BLUE); settextjustify(1,0); setcolor(15); settextstyle(10,0,0);

for(int i = 0; i < 9 ; i++){

bar(20+i\*w+6, 480-10,20+i\*w+w, 480-mf\*a[i]); setusercharsize(w-6,40,mf\*a[i]-10,40);

outtextxy(20+i\*w+(w+14)/2, 470, itos(a[i]));

} }

void Bubble(int a[]){

for(int i = 0 ; i< 9;i++) for(int j = 0;j<9-i-1;j++){

if(a[j]>a[j+1]) {

int x= a[j+1]; a[j+1]=a[j];a[j]=x; cleardevice(); draw(a); getch();

} } }

void Selection(int a[]){

for(int i = 0 ; i< 9;i++){

int low=a[i],pos=i;

for(int j = i+1;j<9;j++) if(low>a[j]) { low = a[j]; pos=j; }

int t = a[i]; a[i]=a[pos]; a[pos]=t;

if(pos!=i){cleardevice();draw(a);getch();}

} }

void Sequential(int a[]){

for(int i = 0 ; i< 9;i++)for(int j = i;j<9;j++){ if(a[j]<a[i]){ int x=a[i]; a[i]=a[j];a[j]=x; cleardevice(); draw(a);getch();

} } }

void Insertion(int a[]){

for(int i = 1 ; i< 9;i++){ int t=a[i],j;

for(j=i-1;j>=0;j--)if(a[j]>t){ a[j+1]=a[j]; cleardevice(); draw(a); getch(); } else break;

a[j+1]=t; cleardevice(); draw(a); getch();

} }

void playSort(){

cleardevice(); char flag = 1,flag2=1;

while(flag){ flag2=1; cleardevice();clrKeyBuff(); settextjustify(1,1); settextstyle(10,0,5); setcolor(LIGHTBLUE); outtextxy(320,25,"Sorting");

Button button[4];

button[0].initialize(25, 100, 615, 170,0,0,"Bubble");button[1].initialize(25, 200,615,270,0,0,"Sequential");

button[2].initialize(25, 300, 615,370,0,0,"Selection");button[3].initialize(25,400, 615,470,0,0,"Insertion");

for(int i = 0; i<4;i++){

button[i].TextColor[NORMAL]=button[i].TextColor[FOCUS]=RED;button[i].Font[NORMAL]=button[i].Font[FOCUS]=4;

button[i].FontSize[NORMAL]=button[i].FontSize[FOCUS]=3; button[i].make();}

int curr=0; button[curr].gotFocus();

while(flag&&flag2){ int k = getch();

switch(k){

case K\_UP: button[curr].lostFocus();

if(--curr==-1) curr=3;

button[curr].gotFocus(); break;

case K\_DOWN: button[curr].lostFocus();

if(++curr==4) curr=0;

button[curr].gotFocus(); break;

case K\_ENTER: int a[] = {2,5,1,7,3,6,9,4,8};

cleardevice(); draw(a); getch();

switch(curr){

case 0: Bubble(a); break; case 1: Sequential(a); break;

case 2: Selection(a); break; case 3: Insertion(a); break;

}; draw(a); getch(); flag2=0; break;

case K\_ESC: flag=0; break;

}; } }; }

#endif

**File: SHM.h**

#ifndef SHM\_H

#define SHM\_H

class Circle{ int x, y, r, c, bc;

public: Circle(int x1, int y1, int rad = 1, int col = 15, int bcol = 0){ x =x1; y=y1; r=rad;c=col;bc=bcol; }

void make() { setcolor(c); circle(x,y,r); }

void erase() { setcolor(bc); circle(x,y,r); }

void set(int x1, int y1) { erase(); x=x1;y=y1; make(); }

};

class Rectangle{ int x, y, l, h, c, bc;

public: Rectangle(int x1, int y1, int hi=1, int le=1, int col=15, int bcol=0){ x=x1;y=y1;h=hi;l=le;c=col;bc=bcol; }

void make(){ setcolor(c); rectangle(x-l/2,y-h/2,x+l/2, y+h/2); }

void erase(){ setcolor(bc); rectangle(x-l/2,y-h/2,x+l/2, y+h/2); }

void set(int x1, int y1) { erase(); x=x1;y=y1; make(); }

};

class Bar{ int x, y, l, h, c, bc;

public:Bar(int x1, int y1, int hi=1, int le=1, int col=15, int bcol=0){ x =x1; y=y1; h=hi;l=le;c=col;bc=bcol; }

void make(){ setfillstyle(1,c); bar(x-l/2,y-h/2,x+l/2, y+h/2); }

void erase(){ setfillstyle(1,bc); bar(x-l/2,y-h/2,x+l/2, y+h/2); }

void set(int x1, int y1){ erase(); x=x1;y=y1; make(); }

};

class Spring{ int x,y,r,ring,h;

public: Spring(int x1, int y1, int r1, int ri, int h1){ x=x1;y=y1;r=r1;ring=ri;h=h1; }

void make(){ setcolor(15); for(int i = y; i<y+h ; i+=h/ring) circle(x,i,r); }

void erase(){ setcolor(0); for(int i = y; i<y+h ; i+=h/ring) circle(x,i,r);}

void sheight(int h1){ erase(); h=h1; make(); }

};

void playSHM(){

cleardevice(); double t=0, a=0.1, b, x; char str[80], temp[80]; gotoxy(14,24);

cout<<"Enter Equation (Carefully): "; cin>>str;

Bar c(350,200,10,10); Circle d(500,204,4); Spring s(150,50,5,20,100); bar(100,45,550,49);

char sm[58]; for(int i = 0; i< 57; i++) sm[i] = i%2+1; sm[57]='\0';

settextstyle(0,0,1); settextjustify(LEFT\_TEXT,CENTER\_TEXT); setcolor(LIGHTBLUE); outtextxy(99,37,sm);

sm[0]=2;sm[1]=0; setcolor(RED); settextjustify(CENTER\_TEXT,CENTER\_TEXT);

outtextxy(150,23,"Spring/Weight"); outtextxy(325,23,"Wire/Block"); outtextxy(500,23,"Pendulum");

while(1){

while(!kbhit()){

setcolor(0); outtextxy(150,213+x,sm); rectangle(315+x/12,50,335-x/12,200+x); line(150,200+x,150,207+x); line(325,200+x,325,207+x); line(500,50,500+x/5,200);

strcpy(temp,str); replace(temp,"t",ftos(t)); t+=a; x=exp(temp); setcolor(15);

rectangle(315+x/12,50,335-x/12,200+x); line(150,200+x,150,207+x); line(325,200+x,325,207+x);

c.set(325,213+x); d.set(500+x/5,204); outtextxy(150,213+x,sm); s.sheight(200+x-50);

line(500,50,500+x/5,200); delay(10);

};

char c = getch(); if(c==' '||c=='p')getch(); else if(c=='r'){playSHM();return;} else if(c==27)return;

} }

#endif

**File: Field.h**

#ifndef FIELD\_H

#define FIELD\_H

void playField(){

cleardevice(); long c[10],x,n,mf=1;

double cx[10], cy[10],r[10], q[10], th=0,e[10],fx,fy, curx = 0, cury =0,fo , k=9000000000 , al, nx, ny, pi=3.14, frequency=1;

gotoxy(1,1); cout<<"\n\tEnter no. of charges: "; cin>>n; cout<<'\n';

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

cout<<"\t\tFor charge "<<i+1<<endl<<"\t\t\tValue: "; cin>>q[i];cout<<"\t\t\tCoordinates (640x480): ";

cin>>cx[i]>>cy[i]; cout<<"\t\t\tLine Color (0-15): "; cin>>c[i];

}

cleardevice(); setcolor(15);

for(i=0;i<n-1;i++) for(int j=i+1;j<n;j++) if(fabs(q[i])<fabs(q[j])){

double t=q[i];

q[i]=q[j];q[j]=t;t=cx[i];cx[i]=cx[j];cx[j]=t;

t=c[i];c[i]=c[j];c[j]=t;t=cy[i];cy[i]=cy[j];cy[j]=t;}

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

if(q[i]>=0){setfillstyle(1,RED);setcolor(RED);} else {setfillstyle(1,GREEN);setcolor(GREEN);}

pieslice(ftoi(cx[i]),ftoi(480-cy[i]),0,360,ftoi(fabs(q[i])\*.9));

} setcolor(0);

for(i = 0 ; i<n;i++){ th=0;

while(!kbhit()&&th<2\*pi){

nx = cx[i] + fabs(q[i])\*cos(th); ny = cy[i] + fabs(q[i])\*sin(th); th+=2\*pi/(fabs(q[i])\*2);

if(q[i]>0) mf=1; else mf=-1;

while(!kbhit()) { curx=nx;cury=ny;fx=fy=0;

for(int j = 0; j<n;j++){

e[j]=k\*mf\*q[j]/(pow((pow(curx-cx[j],2)+pow(cury-cy[j],2)),1.5));

fx+=e[j]\*(curx-cx[j]); fy+=e[j]\*(cury-cy[j]);

}

fo = sqrt(fx\*fx+fy\*fy); while(fo>1)fo/=2;

al = atan2(fy,fx); nx = curx + fo\*cos(al);

ny = cury + fo\*sin(al); int ck=(int(nx\*100)==int(curx\*100)&&int(ny\*100)==int(cury\*100));

if(ck||(!((int)nx==(int)curx&&(int)ny==(int)cury)&&getpixel(nx,480-ny))) break;

putpixel(nx,480-ny,c[i]);

} if(kbhit())if(getch()==27)break;

} }

gotoxy(1,1); cout<<"Done"; getch();

}

#endif

**File: Pong.h**

#ifndef PONG\_H

#define PONG\_H

class Pong{

int BallRadius , ForeColor , BackColor , Key , TempX , TempY[3] , Score[2];

float PaddleSpeed[2]; char Flag , Temp , PositionChanged[2]; char YES, NO, PLAYER, CPU, BALL;

Coord PaddleSize , BallSpeed , Origin , ScreenSize , Position[3];

void initVariables(); void initGame(); void updatePosition(int); void reInitGame(); void drawFig(char, int, int, int);

Label\* PlayerScore; Label\* CPUScore; char\* CPUString; char\* PlayerString;

public: int Play(char\*, int, int, int);

};

void playPong(char name[]){ Pong pong; pong.Play(name,5,RED, LIGHTGREEN); }

int Pong::Play(char\* Name, int HighScore, int FColor = 15, int BColor = 0){

randomize(); ForeColor = FColor; BackColor = BColor; PlayerString = new char[100]; CPUString = new char[100];

PlayerScore = new Label ( 20 , 10 , "" , 0 , 2 , ForeColor , BackColor);

CPUScore = new Label( 400 , 10 , "" , 0 , 2 , ForeColor , BackColor);

strcpy(PlayerString, Name); strcat(PlayerString , "'s score: " ); strcpy(CPUString,"CPU's score: ");

initVariables(); initGame();

while(Flag){ PositionChanged[PLAYER]=PositionChanged[CPU]=NO;

if(kbhit()){ do{ Key=getch();

switch(Key){

case K\_UP: PaddleSpeed[PLAYER]+=1.2; PositionChanged[PLAYER]=1; break;

case K\_DOWN: PaddleSpeed[PLAYER]+=1.2; PositionChanged[PLAYER]=2; break;

case K\_ESC: return 0;

default: continue;

}; break;

}while(1); }

Position[BALL].X += BallSpeed.X; Position[BALL].Y += BallSpeed.Y;

PaddleSpeed[CPU] = random(3) + 2;

if(PaddleSpeed[PLAYER] >= 5) {

if(PositionChanged[PLAYER] == 1) Position[PLAYER].Y -= PaddleSpeed[PLAYER];

else if(PositionChanged[PLAYER] == 2) Position[PLAYER].Y += PaddleSpeed[PLAYER];

if(PositionChanged[PLAYER]){

PaddleSpeed[PLAYER] -= 0.2;PositionChanged[PLAYER] = YES; }

else PaddleSpeed[PLAYER]--;

}

if ( Position[PLAYER].Y <= Origin.Y + PaddleSize.Y / 2 && PositionChanged[PLAYER])

Position[PLAYER].Y = Origin.Y + PaddleSize.Y / 2 + 2;

else if ( Position[PLAYER].Y >= Origin.Y + ScreenSize.Y - PaddleSize.Y / 2 && PositionChanged[PLAYER] )

Position[PLAYER].Y = Origin.Y + ScreenSize.Y - PaddleSize.Y / 2 - 2;

if ( Position[BALL].X <= Origin.X + BallRadius ){

Position[BALL].X = Origin.X + BallRadius;

BallSpeed.X \*= -1; Score[CPU]++; getch();

if ( Score[CPU] == HighScore ) return CPU; reInitGame();

}

else if ( Position[BALL].X >= Origin.X + ScreenSize.X - BallRadius ) {

Position[BALL].X = Origin.X + ScreenSize.X - BallRadius;

BallSpeed.X \*= -1; Score[PLAYER]++; getch();

if ( Score[PLAYER] == HighScore ) return PLAYER; reInitGame();

}

if ( Position[BALL].Y <= Origin.Y + BallRadius ){

Position[BALL].Y = Origin.Y + BallRadius; BallSpeed.Y \*= -1; }

else if ( Position[BALL].Y >= Origin.Y + ScreenSize.Y - BallRadius ) {

Position[BALL].Y = Origin.Y + ScreenSize.Y - BallRadius; BallSpeed.Y \*= -1; }

if ( BallSpeed.X > 0){ if ( Position[CPU].Y < Position[BALL].Y - 10 ) {

Position[CPU].Y += PaddleSpeed[CPU]; PositionChanged[CPU] = YES; }

else if ( Position[CPU].Y > Position[BALL].Y + 10 ) {

Position[CPU].Y -= PaddleSpeed[CPU]; PositionChanged[CPU] = YES; }

}

else if( Position[CPU].Y >= Origin.Y + ScreenSize.Y / 2 + 10){

Position[CPU].Y -= PaddleSpeed[CPU]; PositionChanged[CPU] = YES; }

else if( Position[CPU].Y <= Origin.Y + ScreenSize.Y / 2 - 10){

Position[CPU].Y += PaddleSpeed[CPU]; PositionChanged[CPU] = YES; }

if ( Position[BALL].X <= Position[PLAYER].X + PaddleSize.X / 2 + BallRadius &&

Position[BALL].Y >= Position[PLAYER].Y - PaddleSize.Y / 2 &&

Position[BALL].Y <= Position[PLAYER].Y + PaddleSize.Y / 2 ){

if ( BallSpeed.X < 0 ){

BallSpeed.X \*= -1; Position[BALL].X = Position[PLAYER].X + PaddleSize.X / 2 + BallRadius + 2;

BallSpeed.Y = 5 \* ( ( Position[BALL].Y - Position[PLAYER].Y ) / 25.0 ) + (( Position[BALL].Y > Position[PLAYER].Y ) ? 4 \* PositionChanged[PLAYER] : -4 \* PositionChanged[PLAYER]);

}

}

else if ( Position[BALL].X >= Position[CPU].X - PaddleSize.X / 2 - BallRadius &&

Position[BALL].Y >= Position[CPU].Y-PaddleSize.Y/2 &&

Position[BALL].Y <= Position[CPU].Y+PaddleSize.Y/2){

if( BallSpeed.X > 0 ){

BallSpeed.X \*= -1; Position[BALL].X = Position[CPU].X - PaddleSize.X / 2 - BallRadius - 2;

BallSpeed.Y = 5 \* ( ( Position[BALL].Y-Position[CPU].Y ) / 25.0 ) + ( ( Position[BALL].Y >= Position[CPU].Y ) ? 2 \* PositionChanged[CPU] : -2 \* PositionChanged[CPU]);

}

}

if ( Position[CPU].Y <= Origin.Y + PaddleSize.Y / 2 && PositionChanged[CPU])

Position[CPU].Y = Origin.Y + PaddleSize.Y / 2 + 2;

else if ( Position[CPU].Y >= Origin.Y + ScreenSize.Y - PaddleSize.Y / 2 && PositionChanged[CPU] )

Position[CPU].Y = Origin.Y + ScreenSize.Y - PaddleSize.Y / 2 - 2;

updatePosition(BackColor);

TempX=Position[BALL].X;TempY[2]=Position[BALL].Y;TempY[1]=Position[CPU].Y;

TempY[0]=Position[PLAYER].Y; updatePosition(ForeColor); delay(15);

} getch(); delete PlayerScore; delete CPUScore; return PLAYER;

}

void Pong::updatePosition(int Color){

if(Temp) cleardevice();

drawFig(BALL,TempX,TempY[BALL],Color);

if(PositionChanged[CPU]) drawFig(CPU,Position[CPU].X,TempY[CPU],Color);

if(PositionChanged[PLAYER]) drawFig(PLAYER,Position[PLAYER].X,TempY[PLAYER],Color);

if(Color!=BackColor) rectangle(Origin.X , Origin.Y , Origin.X + ScreenSize.X , Origin.Y + ScreenSize.Y );

}

void Pong::drawFig(char Fig,int x, int y,int Color){

setcolor(Color); setfillstyle(SOLID\_FILL,Color);

if(Fig==BALL){

settextstyle(0,0,2); settextjustify(CENTER\_TEXT,CENTER\_TEXT); outtextxy(x,y,ctos(1)); }

else bar( x - PaddleSize.X / 2 , y - PaddleSize.Y / 2 , x + PaddleSize.X / 2 , y + PaddleSize.Y / 2 );

}

void Pong::initVariables(){

PLAYER = 0; CPU = 1; BALL = 2; YES = 1; NO = 0; PaddleSize.X = 10; PaddleSize.Y = 50; BallSpeed.X = -2;BallSpeed.Y = 2;

BallRadius = textwidth(ctos(1))/2; Key = 0; PaddleSpeed[PLAYER] = PaddleSpeed[CPU] = 5; TempX = 0;

Score[PLAYER] = 0; Score[CPU] = 0; Origin.X = 45; Origin.Y = 40; ScreenSize.X = 550; ScreenSize.Y = 400;

}

void Pong::initGame(){

clrKeyBuff(); cleardevice(); setfillstyle( 1 , BackColor); floodfill( 0 , 0 , ForeColor);

Position[PLAYER].X = Origin.X + 20 ; Position[PLAYER].Y = Origin.Y + 200;

Position[CPU].X = Origin.X + ScreenSize.X - 20;Position[CPU].Y = Origin.Y + ScreenSize.Y - 200;

Position[BALL].X = Origin.X + 275;Position[BALL].Y = Origin.X + 200;

PlayerScore->setText(rstrcat(PlayerString,itos(Score[PLAYER])));

CPUScore->setText(rstrcat(CPUString,itos(Score[CPU]))); TempX = Position[BALL].X;

TempY[PLAYER] = Position[PLAYER].Y; TempY[CPU] = Position[CPU].Y; TempY[BALL]= Position[BALL].Y;

PositionChanged[PLAYER]=PositionChanged[CPU]=YES; updatePosition(ForeColor); getch();

}

void Pong::reInitGame(){ initGame(); }

#endif

**File: Snake.h**

#ifndef SNAKE\_H

#define SNAKE\_H

const char UP =2, RIGHT=1, LEFT=-1, DOWN=-2; struct snakes { Coord pt; char dir; char ch; };

class Snakes{

public: char POINT, SNAKE, WALL, EMPTY,B\_OPEN,B\_CLOSE,B\_DL,B\_DR,B\_UL,B\_UR,DRAW,ERASE,HEAD,TAIL;

int pix, score, length,backcolor,level; Coord point; snakes snake[500];

char key, del, direction, moving, speed, flag, screen[50][66]; Label\*l;

Snakes(); void play(); void initialize(); void putFigure(Coord, char, char=1, char=0); void updateSnake(); void checkInput();

void setScreen(int); void changeDir(int); void tab(int); void chooseSnake(); void resetPoint(); void printScreen();

void lost(char); char exist(int,int); char exist(Coord);

};

void playSnake(){ Snakes snake; snake.chooseSnake(); snake.play(); }

Snakes::Snakes(){ level = -1; l = new Label(550,10,""); initialize(); }

void Snakes::changeDir(int dir){ direction = dir; }

char Snakes::exist(int x,int y){ return screen[y][x]; }

void Snakes::lost(char why){ flag = why; }

void Snakes::checkInput(){

if(kbhit()){ key = getch();

switch(key) {

case K\_UP: changeDir(UP); break;

case K\_DOWN: changeDir(DOWN); break;

case K\_LEFT: changeDir(LEFT); break;

case K\_RIGHT: changeDir(RIGHT); break;

case 'p': getch();

case 0: checkInput(); break;

case K\_ESC: flag = 21; break;

} } }; }

void Snakes::resetPoint(){

pt: point.X = random(60)+2; point.Y = random(45)+2; if(exist(point)) goto pt;

screen[point.Y][point.X]=POINT; putFigure(point,POINT);

}

char Snakes::exist(Coord coord){ return screen[coord.Y][coord.X]; }

void Snakes::initialize(){

randomize(); clrKeyBuff(); snake[0].pt.X = 5; snake[0].pt.Y = 12; snake[0].dir = direction = RIGHT; length = 1;

score = 1; speed = 1; del=0; flag = 0; pix = 10; backcolor=0; level++; resetPoint(); B\_OPEN = 5; B\_CLOSE = 6;

POINT = 20; DRAW = 1; ERASE = 0; WALL = -1; EMPTY = 0;DRAW = 1;

}

void Snakes::chooseSnake()

{

cleardevice();settextstyle(4,0,5);setcolor(2);settextjustify(CENTER\_TEXT,CENTER\_TEXT);

outtextxy(300,50,"Design Your Snake"); settextstyle(0,0,1); setcolor(4);

for(int i=0;i<5;i++) outtextxy((11+i\*10)\*pix-5,-5+21\*pix-1.5\*pix,itos(i+1));

Coord pos = {11,21}; putFigure(pos,2); pos.X+=10; putFigure(pos,6,1,RIGHT); pos.X+=10; putFigure(pos,18);

pos.X+=10; putFigure(pos,12,1,RIGHT); pos.X+=10; putFigure(pos,15); int a=0; gotoxy(20,17);

cout<<"Enter Head Character: "; cin>>a;

if(++a==3)a=5; else if(a==4)a=18; else if(a==5)a=12;else if(a==6)a=15; HEAD=a;

gotoxy(20,18); cout<<"Enter Snake Character: "; cin>>a;

if(++a==3)a=6; else if(a==4)a=18; else if(a==5)a=12; else if(a==6)a=15;

gotoxy(20,19); cout<<"Enter Tail Character: "; SNAKE=a;

cin>>a; if(++a==3)a=5; else if(a==4)a=18; else if(a==5)a=12; else if(a==6)a=15; TAIL=a;

snake[0].ch = SNAKE; cleardevice();

}

void Snakes::setScreen(int value = 0){

switch(value){

case 1: setScreen(0);

for(int i = 1; i<65;i++){

screen[1][i] = WALL; screen[48][i] = WALL;

if(i<49){ screen[i][1] = WALL; screen[i][64] = WALL; }

} break;

case 0: for(i = 0; i<50;i++) for(int j = 0;j<66;j++) screen[i][j] = EMPTY; break;

case 2: setScreen(0);

for(i = 1; i<65;i++){

if(!b\_w(i,32,40))screen[22][i] = WALL;

screen[26][i] = WALL; if(b\_w(i,9,19))screen[9][i] = WALL;

if(b\_w(i,0,22)){ screen[i][1] = WALL; screen[i][24] = WALL; }

if(b\_w(i,26,49)) screen[i][24] = WALL;

} break;

} }

void Snakes::printScreen(){

for(int i = 1;i<49;i++)for(int j = 1;j<65;j++){ Coord x = {j,i}; putFigure(x,screen[i][j]); }

}

void Snakes::updateSnake(){ if(!del) {

screen[snake[length-1].pt.Y][snake[length-1].pt.X] = EMPTY;

screen[snake[length-2].pt.Y][snake[length-2].pt.X] = TAIL;

putFigure(snake[length-1].pt, TAIL, ERASE,snake[length-1].dir);

putFigure(snake[length-2].pt, snake[length-2].ch, ERASE,snake[length-2].dir);

if(TAIL==B\_OPEN) TAIL=B\_CLOSE;

else if(TAIL ==B\_CLOSE) TAIL=B\_OPEN;

putFigure(snake[length-2].pt, TAIL,DRAW,snake[length-2].dir);

snake[length-1].pt.X = snake[length-1].pt.Y = 0; }

else{ del--; length++; }

for(int i = length-1;i>=0;i--) snake[i+1] = snake[i];

if(abs(direction) == 1) snake[0].pt.X += direction;

else snake[0].pt.Y -= direction/2;

putFigure(snake[1].pt, HEAD, ERASE,snake[1].dir);

snake[0].ch = SNAKE; snake[0].dir = direction;

if(snake[0].pt.X < 0) snake[0].pt.X = 65;

else if(snake[0].pt.X > 65) snake[0].pt.X = 0;

if(snake[0].pt.Y <0) snake[0].pt.Y = 48;

else if(snake[0].pt.Y>49) snake[0].pt.Y = 0;

char newhead = screen[snake[0].pt.Y][snake[0].pt.X];

if(newhead ==WALL) { lost(WALL); return; }

else if(b\_w(newhead,0,20)){ lost(SNAKE); return; }

else if(newhead>=20){ del = 3; resetPoint(); speed++; l->setText(rstrcat("Score: ",itos(score-1)));

putFigure(snake[0].pt,POINT,ERASE); }

putFigure(snake[0].pt, HEAD,DRAW,snake[0].dir); putFigure(snake[1].pt, snake[1].ch,DRAW,snake[1].dir);

screen[snake[1].pt.Y][snake[1].pt.X] = SNAKE; screen[snake[0].pt.Y][snake[0].pt.X] = HEAD;

}

void Snakes::play()

{

initialize(); cleardevice(); setScreen(level-1); printScreen(); resetPoint(); del=3; l->setText("Score: 0");

flag=0; checkInput();

while(!flag){

if(score%6==0){ resetPoint(); score++; }

updateSnake(); if(speed<11) delay(100-6\*speed);

else {play();break;}; checkInput();

} score = score + 10\*(level-1) - 1;

}

void Snakes::putFigure(Coord coord, char character, char type, char dir)

{

int ox=-5,oy=-5;

switch(character){

case -1: setfillstyle(1,1);

bar(coord.X\*pix-pix/2+ox,coord.Y\*pix-pix/2+oy,coord.X\*pix+pix/2+ox,coord.Y\*pix+pix/2+oy);

break;

case 20: if(type==ERASE) setcolor(backcolor);

else setcolor(4);

settextjustify(CENTER\_TEXT,CENTER\_TEXT); settextstyle(0,0,1);

outtextxy(coord.X\*pix+ox,oy+ coord.Y\*pix, ctos((char)1)); break;

case 18:if(type==ERASE)setcolor(backcolor);

else setcolor(3);

settextjustify(CENTER\_TEXT,CENTER\_TEXT);

settextstyle(0,0,1);

outtextxy(coord.X\*pix+ox,oy+ coord.Y\*pix, ctos('x'));

break;

case 2: if(type==ERASE) { setcolor(backcolor); setfillstyle(1,backcolor); }

else{ int c=random(13)+1;setcolor(c); setfillstyle(1,c); }

pieslice(coord.X\*pix+ox, coord.Y\*pix+oy, 0, 360, (pix)/2); break;

case 5: if(type==ERASE)setcolor(backcolor); else setcolor(15);

settextjustify(1,1); setusercharsize(1,2,1,2); if(dir%2) settextstyle(1,1,0);

else settextstyle(1,0,0); outtextxy(ox+coord.X\*pix,oy+coord.Y\*pix,"("); break;

case 6: if(type==ERASE) setcolor(backcolor);

else setcolor(15); settextjustify(CENTER\_TEXT,CENTER\_TEXT);

setusercharsize(1,2,1,2); if(dir%2) settextstyle(1,1,0); else settextstyle(1,0,0);

outtextxy(ox+coord.X\*pix,oy+coord.Y\*pix,")"); break;

case 12: setcolor(0);

if(type == ERASE) setfillstyle(1,0); else setfillstyle(1,15); int s1=0,e1=0,s2=0,e2=0;

if(dir==LEFT){s1=180+20;e1=360;s2=0;e2=180-20;}else if(dir==RIGHT){s1=0+20;e1=360-20;s2=e2=0;}

else if(dir==UP){s1=90+20;e1=360;s2=0;e2=90-20;}else if(dir==DOWN){s1=0;e1=270-20;s1=270+20;e2=360;}

pieslice(coord.X\*pix+ox,coord.Y\*pix+oy,s1,e1,pix/2+1); pieslice(coord.X\*pix+ox,coord.Y\*pix+oy,s2,e2,pix/2+1);

break;

case 15:if(type==ERASE) setfillstyle(1,0);

else setfillstyle(1,2);

bar(coord.X\*pix-pix/2+ox+1,coord.Y\*pix-pix/2+oy+1,coord.X\*pix+pix/2+ox-1,coord.Y\*pix+pix/2-1+oy); break;

} }

#endif

**File: Graph.h**

#ifndef GRAPH\_H

#define GRAPH\_H

double graph(char fn[],char inv=0,double scaleX=1,double scaleY=1,double accuracy=1,double sDomain = 0,double

eDomain = 0,double width = 640,double height = 480,char minus =0,char color = WHITE, char m = 0){

char temp[1000],fn2[1000]; temp[0] = '\0'; strcpy(fn2,fn); replace(fn2,"x","(x)");

double frequency = 1.0/accuracy;

if((sDomain\*scaleX)<-1\*width/2) sDomain=-1\*width/(2\*scaleX);

if((eDomain\*scaleX)>width/2) eDomain=width/(scaleX\*2);

if(sDomain==eDomain&&sDomain==0){ sDomain=-1\*width/(2\*scaleX); eDomain=width/(2\*scaleX); }

double x = sDomain, y = 0, tvalx = 0, tvaly=0, tvalx1=0, tvaly1=0, sum=0;

while(!kbhit()&&x<=eDomain){

strcpy(temp, fn2); char \*xs = new char[20]; sprintf(xs,"%lf",x); replace(temp, "x" , xs); delete xs;

y = exp(temp); sum+=fabs(y\*frequency); if(m)y\*=-1;

if(ftoi(x\*scaleX+width/2)==tvalx && tvaly==ftoi(height/2-y\*scaleY)){

x+=frequency;continue; }

else { tvalx1 = ftoi(x\*scaleX+width/2); tvaly1 = ftoi(height/2-y\*scaleY); }

if(b\_w(y\*scaleY,-1\*height/2-1,height/2+1)&&b\_w(x\*scaleX,-1\*width/2-1,width/2+1)&&tvalx&&tvaly)

if(!inv){ line(tvalx,tvaly,tvalx1,tvaly1); if(minus) line(tvalx,480-tvaly,tvalx1,480-tvaly1);}

else { putpixel(ftoi(width/2+y\*scaleY),ftoi(-x\*scaleX+height/2),color);

if(minus)putpixel(ftoi(width/2+y\*scaleY),ftoi(x\*scaleX+height/2),color); }

x+=frequency; tvalx=tvalx1;tvaly=tvaly1;

} temp[0] = '\0'; clrKeyBuff();

return sum;

}

int matherr(struct exception \*a){ a->retval=0; return 5; }

int showGraph(char \* fn){

double scaleX =1, scaleY=1, accuracy=10, sDomain = 0, eDomain = 0;

char minus =0, inverse =0, faa=1; int curr = 0; Button button[8]; TextBox tbox[7];

do{ cleardevice();

button[0].initialize(500, 315, 620, 330,0,0,"Scale - X"); button[1].initialize(500, 335, 620, 350,0,0,"Scale - Y");

button[2].initialize(500, 355, 620, 370,0,0,"Start Range");button[3].initialize(500, 375, 620, 390,0,0,"End Range");

button[4].initialize(500, 395, 620, 410,0,0,"Accuracy");button[5].initialize(500, 415, 620, 430,0,0,"Invresion");

button[6].initialize(500, 435, 620, 450,0,0,"Added (+-)");button[7].initialize(500, 455, 620, 470,0,0,"Function");

for(int i = 0; i<8;i++){

button[i].BorderColor[NORMAL] = 0; button[i].BorderSize[FOCUS] = 1; button[i].make(); }

button[curr].gotFocus(); settextjustify(LEFT\_TEXT, TOP\_TEXT);

outtextxy(490, 40, "Scale"); outtextxy(490, 120, "Range"); outtextxy(510, 60, "X-axis:"); outtextxy(510, 80, "Y-axis:");

outtextxy(510, 140, "Start :");outtextxy(510, 160, "End :");outtextxy(495, 200, "Accuracy:");

outtextxy(495, 230, "Inverse :");outtextxy(495, 260, "Add (+-):");outtextxy(490, 300, "Change");

tbox[0].initialize(570,60,8,ftos(scaleX));tbox[1].initialize(570,80,8,ftos(scaleY));tbox[2].initialize(570,140,8,ftos(sDomain);

tbox[3].initialize(570,160,8,ftos(eDomain));tbox[4].initialize(570,200,8,ftos(accuracy));

tbox[5].initialize(570,230,8,itos(inverse));tbox[6].initialize(570,260,8,itos(minus));

graph("0",0,1,1,1,0,0,480);graph("0",1,1,1,1,0,0,480);graph("239",1,1,1,1,0,0,480);

double area = graph(fn,inverse,scaleX, scaleY, accuracy, sDomain, eDomain, 480, 480, minus);

gotoxy(1,1); cout<<"Area under graph = "<<area; faa=1;

while(faa){ int k = getch();

switch(k){

case K\_UP: button[curr].lostFocus();

if(--curr==-1) curr=7;

button[curr].gotFocus(); break;

case K\_DOWN: button[curr].lostFocus();

if(++curr==8) curr=0;

button[curr].gotFocus(); break;

case K\_ENTER: if(curr==7) return 1;

tbox[curr].setText(""); tbox[curr].focus(); char t[100]; strcpy(t, tbox[curr].text);

switch(curr){

case 0: scaleX=stof(t); break;

case 1: scaleY=stof(t); break;

case 2: sDomain=stof(t); break;

case 3: eDomain=stof(t);break;

case 4: accuracy=stof(t);break;

case 5: inverse=stoi(t); break;

case 6: minus=stoi(t); break;

} faa=0;break;

case K\_ESC: return 0;

} }

}while(1);

}

void playGraph(){

cleardevice();char fn[100];

while(1){

gotoxy(10, 12);cout<<"Enter the Equation: ";cin>>fn;

if(showGraph(fn))continue; else return;

} }

#endif

**File Tic.h**

#ifndef TIC\_H

#define TIC\_H

const char CIRCLE = -1, CROSS = 1, TEMP= 0, PERMANENT= 1, ERASE= -1, ENTER = 5;

coord origin = {0, 0}; char curr = 0; int box = 100; char linewidth, tic[9];

void drawit(){

line(origin.x + box , origin.y ,origin.x + box , origin.y + box \* 3);

line(origin.x + box \* 2 , origin.y , origin.x + box \* 2 , origin.y + box \* 3);

line(origin.x , origin.y + box ,origin.x + box \* 3 , origin.y + box);

line(origin.x , origin.y + box \* 2 , origin.x + box \* 3 , origin.y + box \* 2);

}

void drawFig(char cr, char fig, char type = PERMANENT){

int color = getcolor(); coord co = {cr%3+1,cr/3+1};

if(type==TEMP) setcolor(RED);

else if(type==PERMANENT) setcolor(WHITE);

else if(type==ERASE){ if(tic[cr]) goto here;

setcolor(BLACK);}

if(fig == CROSS) {

line(origin.x + box\*(co.x + 0.2 -1), origin.y+box \* (co.y + 0.2-1), origin.x + box\*(co.x-0.2), origin.y+box\*(co.y-0.2));

line(origin.x + box\*(co.x + 0.2 -1), origin.y+box \* (co.y - 0.2), origin.x + box\*(co.x-0.2), origin.y+box\*(co.y+0.2-1));

}

else if(fig == CIRCLE)circle(origin.x + box \* (co.x + 0.5 - 1), origin.y + box \* (co.y + 0.5 - 1) , 0.4 \* box);

here: setcolor(color);

}

int getNo(int n){

int x, o,nx=0,no=0,num=0;

if(n<0){ o = n/CIRCLE; x = 0; }

else{ x = n/CROSS; o = 0; }

for(int i = 0; i<9 ; i++){

if(tic[i] == CROSS) nx++;

else if(tic[i] == CIRCLE) no++;

if(!((i+1)%3)){ if (nx == x && no == o) num++;

nx=no=0;

} }

for(int j=0;j<3;j++){

for(i=j;i<9;i+=3)

if(tic[i]==CROSS)nx++;

else if(tic[i]==CIRCLE)no++;

if(nx==x&&no==o)num++; nx=no=0;

}

for(i=0;i<9;i+=4) {

if(tic[i] ==CROSS) nx++;

else if(tic[i] ==CIRCLE) no++;

}

if(nx==x&&no==o) num++; nx=no=0;

for(i=2;i<8;i+=2) if(tic[i]==CROSS) nx++;

else if(tic[i]==CIRCLE) no++;

if(nx==x&&no==o) num++;

return num;

}

char checkWin(){ if(getNo(3\*CIRCLE)) return CIRCLE; else if(getNo(3\*CROSS)) return CROSS; else return 0; }

int getStateValue(){

if(getNo(3\*CIRCLE)) return 50;

else if(specialCases()) return -50;

else{ int x2 = getNo(2\*CROSS), x1 = getNo(CROSS), o2 = getNo(2\*CIRCLE), o1 = getNo(CIRCLE);

return 3\*o2 + o1 - (3\*x2 + x1); }

}

char flag=1; char cfig=CROSS; int key=0,moves=0;

void changePos(char k){

if(k==ENTER){ tic[curr]=cfig; drawFig(curr,cfig,PERMANENT); cfig\*=-1; k=10; }

do{

if(!checkWin()&&k!=10){ drawFig(curr,cfig,ERASE);

if(abs(k)==1) curr+=k; else if(k==UP) curr-=2+k/2; else curr+=2-k/2;

if(curr>=9)curr=curr-9; else if(curr<=-1)curr=9+curr; if(tic[curr]) continue;

drawFig(curr,cfig,TEMP);

}break;

}while(1);

}

void playTicTacToe(){

flag=1;origin.x = 150;origin.y = 75;curr = 0;box = 100;cfig=CROSS;key=0;moves=0;randomize();cleardevice();

settextstyle(10,0,2);setcolor(LIGHTBLUE); outtextxy(100,40,"Tic Tac Toe"); linewidth = 3;

setlinestyle(0,0,linewidth); box += 4\*linewidth; drawit(); int CPUMove = random(2);

for(int i =0;i<9;i++) tic[i]=0;

while(flags) {

if(CPUMove){

int highvalue = -50, bestposition = 0;

for(i = 0;i<9;i++) { if(!tic[i]){

drawFig(i, CIRCLE, TEMP); tic[i] = CIRCLE;

int statevalue = getStateValue();

if(statevalue > highvalue){ highvalue = statevalue; bestposition = i; }

else if(statevalue == highvalue) highvalue = (random(2))?statevalue:highvalue;

delay(10); tic[i] = 0; drawFig(i, CIRCLE, ERASE);

} }

drawFig(bestposition, CIRCLE, PERMANENT); tic[bestposition] = CIRCLE; moves++;

}

if(checkWin()==CIRCLE) outtextxy(380,420,"Computer wins"); flag=0;

else if(moves == 9) outtextxy(380,420,"Its a Draw"); flag=0;

CPUMove = 1; int moved = 0; cfig = CROSS; changePos(RIGHT); cfig = CROSS;

if(flag) do{ key = getch();

switch(key){

case K\_UP: changePos(UP); cfig=CROSS; break;

case K\_DOWN: changePos(DOWN); cfig=CROSS; break;

case K\_RIGHT: changePos(RIGHT); cfig=CROSS; break;

case K\_LEFT: changePos(LEFT); cfig=CROSS; break;

case K\_ESC: flag=0; break;

case K\_ENTER: changePos(ENTER); moved = 1; break;

default: continue;

};

}while(!moved); moves++;

if(flag) if(checkWin()==CROSS) outtextxy(380,420,"Player wins");flag=0;

else if(moves == 9)outtextxy(380,420,"Its a Draw"); flag=0;

}getch();

}

#endif