پروژه مبانی برنامه نویسی

برنامه Block Puzzle

سید علیرضا میرعابدینی کچومثقالی ، مبین ولی پناه ، محمدرضا نوروزی

## تابع ConXY

void ConXY(int x, int y)

{

    COORD coord;

    coord.X = x;

    coord.Y = y;

    SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

    return;

}

کد تابع ConXY

این تابع برای تغییر دادن مکان شناور در کنسول نوشته شده که با استفاده از توابع داخل کتابخانه windows.h این کار را انجام می‌دهد

## تابع ConClr

void ConClr(int clr)

{

    SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), clr);

    return;

}

کد تابع ConClr

این تابع برای تغییر رنگ متن در نوشتار شناور کنسول تعریف شده که با استفاده از توابع موجود در کتابخانه windows.h این کار را انجام می‌دهد.

کلاس Game

این کلاس، بخش اصلی کد ما هست و عملا کل کد بازی در این بخش هست.

متغییر های داخل کلاس

    bool easy\_Single[4][4] = {{true, false, false, false}, {false, false, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool easy\_Bar\_1[4][4] = {{true, true, false, false}, {false, false, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool easy\_Bar\_2[4][4] = {{true, false, false, false}, {true, false, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool easy\_Square[4][4] = {{true, true, false, false}, {true, true, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Norm\_L\_1[4][4] = {{false, true, false, false}, {true, true, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Norm\_L\_2[4][4] = {{true, false, false, false}, {true, true, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Norm\_L\_3[4][4] = {{true, true, false, false}, {false, true, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Norm\_L\_4[4][4] = {{true, true, false, false}, {true, false, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Norm\_Bar\_3\_1[4][4] = {{true, false, false, false}, {true, false, false, false}, {true, false, false, false}, {false, false, false, false}};

    bool Norm\_Bar\_3\_2[4][4] = {{true, true, true, false}, {false, false, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_LL\_1[4][4] = {{true, true, false, false}, {false, true, true, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_LL\_2[4][4] = {{false, true, true, false}, {true, true, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_LL\_3[4][4] = {{true, false, false, false}, {true, true, false, false}, {false, true, false, false}, {false, false, false, false}};

    bool Hard\_LL\_4[4][4] = {{false, true, false, false}, {true, true, false, false}, {true, false, false, false}, {false, false, false, false}};

    bool Hard\_T\_1[4][4] = {{true, true, true, false}, {false, true, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_T\_2[4][4] = {{false, true, false, false}, {true, true, true, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_T\_3[4][4] = {{false, true, false, false}, {true, true, false, false}, {false, true, false, false}, {false, false, false, false}};

    bool Hard\_T\_4[4][4] = {{true, false, false, false}, {true, true, false, false}, {true, false, false, false}, {false, false, false, false}};

    bool Hard\_L\_1[4][4] = {{true, true, true, false}, {true, false, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_L\_2[4][4] = {{true, true, true, false}, {false, false, true, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_L\_3[4][4] = {{true, false, false, false}, {true, true, true, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_L\_4[4][4] = {{false, false, true, false}, {true, true, true, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_Bar\_4\_1[4][4] = {{true, true, true, true}, {false, false, false, false}, {false, false, false, false}, {false, false, false, false}};

    bool Hard\_Bar\_4\_2[4][4] = {{true, false, false, false}, {true, false, false, false}, {true, false, false, false}, {true, false, false, false}};

متغیر های شکل بلوک ها

|  |  |  |  |
| --- | --- | --- | --- |
| T | F | F | F |
| T | T | F | F |
| T | F | F | F |
| F | F | F | F |

این بخش از متغییر ها مختص به تعریف بلوک ها به شکل یک ماتریس از نوع بولین هست که به جای فضاهای هر بلوک درست و خالی نادرست است.



-------->

T:True F:False

برای مثال یکی ار اشکال که در متغییر Hard\_T\_1 تعریف شده

    bool Board[100][100] = {false};

    bool Tmp[100][100] = {false};

    int cord = 0;

    unsigned int Score = 0;

    int boardsize[2] = {11, 11};

    unsigned int HighScore = 0;

    int diff[3] = {1, 1, 1};

متغییر های دیگر

آرایهBoard Boolean

این آرایه تخته بازی هست که در ان مکان های پر شده true و بقیه جاها false هستند و برای بعد اول این آرایه برای طول و بعد دوم برای عرض صفحه بازی در نظر گرفته شده و بدیهی است فقط خانه های لازم در آرایه همیشه استفاده میشوند.

آرایه Boolean TMP

این آرایه برای پردازش بر روی صفحه بازی بدون دستکاری آرایه Board درست شده.

متغییر Integer cord

مختصات طول گوشه سمت چپ بلوک در حال حرکت در آن ذخیره میشود

آرایهInteger boardsize

طول طول و عرض صفحه بازی را در آن ذخیره میشود

متغییر های Integer Score و HighScore

در آنها به ترتیب امتیاز لحظه ای فرد و بیشترین امتیاز فرد از موقع باز کردن برنامه نمایش داده می‌شود.

آرایه Integer diff

در این آرایه ضریب احتمال هر دسته مهره مشخص میشود که در بخش تابع RndSel بیشتر توضیح خواهیم داد.

توابع کلاس

توابع خصوصی

تابع void Round\_up

    int Round\_up(double in)

    {

        if (in == int(in))

            return in;

        return (1 + in);

    }

کد تابع Round\_up

این تابع وظیفه عمگر رند کردن رو به بالا اعداد را دارد

تابع integer RndSel

    int RndSel()

    {

        srand(time(0));

        int asd = 4 \* diff[0] + 8 \* diff[1] + 12 \* diff[2];

        int opt = 1 + (rand() % asd);

        if (opt <= 4 \* diff[0])

            return Round\_up(opt / diff[0]);

        opt -= 4 \* diff[0];

        if (opt <= 8 \* diff[1])

            return Round\_up(4 + (opt / diff[1]));

        opt -= 8 \* diff[1];

        return Round\_up(12 + (opt / diff[2]));

    }

کد تابع RndSel

ابتدا با توجه به وزن شانسی که در برای هر بلوک یا توجه به درجه سختی در نظرگرفته وزن شانس کل را حساب کرده و در متغییر asd ذخیره می‌کند. سپس یک عدد تصادفی بین ۱ تا وزن شانس کل در نظر می‌گیرد و در متغییر opt ذخیره می‌کند.اگر خروجی از وزن شانس کل بلوک های ساده کمتر باشد به این معناست که بلوک انتخابی از بلوک های ساده هست و باتوجه به اعداد انتخابی برای هر بلوک اگر خروجی رو تقسیم بر وزن شانس هر بلوک ساده کنیم ، شماره بلوک ساده بدست می‌آید. اگر خروجی بین وزن شانس کل بلوک های ساده و مجموع وزن شانس بلوک های متوسط و ساده باشد یعنی بلوک انتخابی از بلوک های متوسط بوده و از این خروجی وزن شانس بلوک های ساده را کم کرده و تقسیم بر مقدار خروجی کرده و بعلاوه تعداد بلوک های ساده میکنیم تا شماره بلوک مورد نظر را بدست می‌آوریم و همین کار را برای حالت بلوک های سخت انجام می‌دهیم.

تابع void Render\_Game\_Board

    void Render\_Game\_Board(bool board[100][100])

    {

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

        {

            for (int j = 0; j < boardsize[1]; j++)

            {

                ConXY(2 \* i + 2, j + 1);

                if (board[i][j])

                    cout << '#';

                else

                    cout << '-';

            }

        }

    }

کد تابع Render\_Game\_Board

این تابع وظیفه تغییر صفحه بازی یا توجه به صفحه داده شده به آن را دارد.

تابع void Copy\_Board\_To\_Tmp

    void Copy\_Board\_To\_Tmp()

    {

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

        {

            for (int j = 0; j < boardsize[1]; j++)

                Tmp[i][j] = Board[i][j];

        }

    }

کد تابع Copy\_Board\_To\_Tmp

این تابع وظیفه کپی کردن صفحه بازی در این متغییر دارد.

تابع void Clear\_a\_Row

    void Clear\_a\_Row()

    {

        for (int i = 0; i < boardsize[1]; i++)

        {

            for (int j = 0; j < boardsize[0]; j++)

            {

                if (Board[j][i])

                {

                    for (int k = 0; k < boardsize[0]; k++)

                    {

                        Board[k][i] = false;

                    }

                    return;

                }

            }

        }

    }

کد تابع Clear\_a\_Row

این تابع وظیفه دارد آخرین سطر جدول بازی را پاک کند که فراخوانی آن فقط در شرایط امتیاز بالای ۵۰ امکان پذیر است

تابع Void Hint

    void Hint(int x, int y)

    {

        ConClr(226);

        ConXY(x, y);

        cout << "UP & DOWN ARROW : SELECTING ITEMS IN MENU";

        ConXY(x, y + 1);

        cout << "         A : MOVE BLOCK TO LEFT          ";

        ConXY(x, y + 2);

        cout << "         D : MOVE BLOCK TO RIGHT         ";

        ConXY(x, y + 3);

        cout << "         S : FALL DOWN THE BLOCK         ";

        ConXY(x, y + 4);

        cout << "                                         ";

        ConXY(x, y + 5);

        cout << "        PRESS ANY KEY TO CONTINUE        ";

        ConClr(7);

        \_getch();

        for (int i = 0; i < 6; i++)

        {

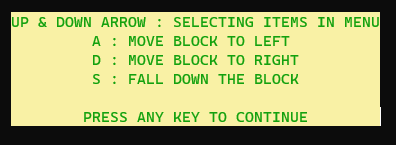
            ConXY(x, y + i);

            cout << "                                         ";

        }

    }

کد تابع Hint

این تابع وظیفه چاپ راهنمای کلید های بازی را دارد که بعد از فشردن کلید توسط کاربر آن را پاک می‌کند.

نمونه خروجی این تابع

تابع Void High\_Score

    void High\_Score(int x, int y)

    {

        ConClr(186);

        ConXY(x, y);

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

        ConXY(x, y + 1);

        cout << " |  \_\_ \\|  \_\_\_\_/ \_\_\_\_/ \_\_ \\|  \_\_ \\|  \_\_ \\| | ";

        ConXY(x, y + 2);

        cout << " | |\_\_) | |\_\_ | |   | |  | | |\_\_) | |  | | | ";

        ConXY(x, y + 3);

        cout << " |  \_  /|  \_\_|| |   | |  | |  \_  /| |  | | | ";

        ConXY(x, y + 4);

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

        ConXY(x, y + 5);

        cout << " |\_|  \\\_\\\_\_\_\_\_\_\\\_\_\_\_\_\\\_\_\_\_/|\_|  \\\_\\\_\_\_\_\_/(\_) ";

        ConXY(x, y + 6);

        cout << "                                             ";

        ConXY(x, y + 7);

        cout << "          PRESS ANY KEY TO CONTINUE          ";

        ConClr(7);

        \_getch();

        for (int i = 0; i < 8; i++)

        {

            ConXY(x, y + i);

            cout << "                                             ";

        }

    }

کد تابع High\_Score

این تابع وظیفه چاپ پیام رکورد بازی را دارد که بعد از فشردن کلید توسط کاربر آن را پاک می‌کند.

نمونه خروجی این تابع

تابع Void Exit

    void Exit()

    {

        system("cls");

        ConClr(10);

        cout << "oooooooooo o888                        oooo         oooooooooo                                      o888             \n 888    888 888   ooooooo     ooooooo   888  ooooo   888    888 oooo  oooo  ooooooooooo ooooooooooo  888  ooooooooo8 \n 888oooo88  888 888     888 888     888 888o888      888oooo88   888   888       8888        8888    888 888oooooo8  \n 888    888 888 888     888 888         8888 88o     888         888   888    8888        8888       888 888         \no888ooo888 o888o  88ooo88     88ooo888 o888o o888o  o888o         888o88 8o o888ooooooo o888ooooooo o888o  88oooo888 \n ";

        ConClr(7);

        ConXY(50, 16);

        ConClr(4);

        cout << "High Score : " << HighScore;

        ConXY(50, 15);

        cout << "Last Score : " << Score;

        ConClr(109);

        ConXY(47, 18);

        cout << "PRESS ANY KEY TO EXIT";

        \_getch();

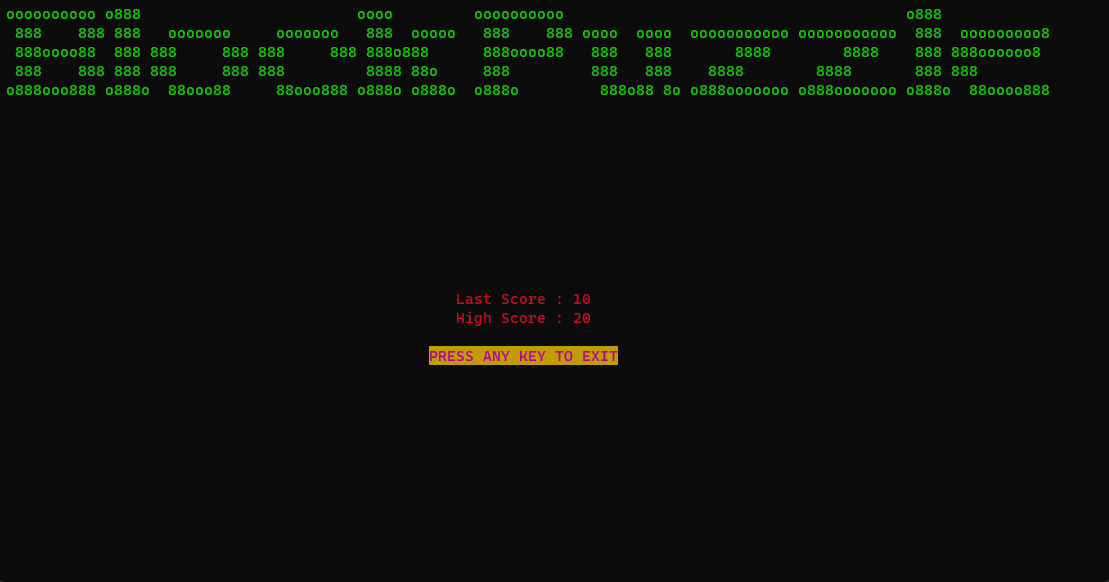
        ConClr(7);

        system("cls");

        exit(0);

    }

کد تابع Exit

این تابع صفحه خروج از بازی با نمایش آخرین و بالاترین امتیاز کاربر را نشان میدهد و کاربر با فشردن کلیدی بازی را ترک میکند

نمونه خروجی این تابع

تابع Void Make\_Game\_Board

    void Make\_Game\_Borad()

    {

        ConClr(238);

        ConXY(0, 0);

        for (int i = 1; i <= 1 + 2 \* (1 + boardsize[0]); i++)

        {

            cout << " ";

        }

        ConXY(0, (boardsize[1] + 1));

        for (int i = 1; i <= 1 + 2 \* (1 + boardsize[0]); i++)

        {

            cout << " ";

        }

        for (int i = 1; i <= boardsize[1]; i++)

        {

            ConXY(0, i);

            cout << " ";

            ConXY((1 + boardsize[0]) \* 2, i);

            cout << " ";

        }

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

        {

            for (int j = 0; j < boardsize[1]; j++)

            {

                ConXY(2 \* i + 2, j + 1);

                cout << "-";

            }

        }

    }

کد تابع Make\_Game\_board

این تابع وظیفه چاپ یک صفحه بازی خالی با ستون را دارد.

تابع void change\_Game\_Board

    void change\_Game\_Board(int X, int x, int y, bool obj[4][4])

    {

        int Y = -1;

        for (int i = 0; i <= boardsize[1] - y; i++)

        {

            for (int j = 0; j < y; j++)

            {

                for (int k = 0; k < x; k++)

                {

                    if (obj[k][j])

                    {

                        if (Board[X + k][i + j])

                        {

                            Y = i - 1;

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

                            {

                                for (int k = 0; k < x; k++)

                                {

                                    if (obj[k][i])

                                        Board[X + k][Y + i] = true;

                                }

                            }

                            Render\_Game\_Board(Board);

                            return;

                        }

                    }

                }

                if (Y > 0)

                    break;

            }

            if (Y > 0)

                break;

        }

        if (Y == -1)

        {

            Y = boardsize[1] - y;

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

            {

                for (int k = 0; k < x; k++)

                {

                    if (obj[k][i])

                        Board[X + k][Y + i] = true;

                }

            }

            Render\_Game\_Board(Board);

        }

        return;

    }

کد تابع change\_Game\_Board

این تابع با توجه به شکل جسم،ابعاد جسم و تخته تا جایی که در مکان قرارگیری جسم پر نشده باشد پیش می‌رود و درصورت برخورد جسم را در یه پله بالاتر که میتوانسته قرار بگیرد ، قرار میدهد و اگر برخرد نداشت شکل را در پایین‌ترین خانه ممکن قرار می‌دهد و بعد صفحه را باز آرایی می‌کند.

تابع Void Do

    void Do(int cor, int mod)

    {

        switch (mod)

        {

        case 1:

            change\_Game\_Board(cor, 1, 1, easy\_Single);

            break;

        case 2:

            change\_Game\_Board(cor, 2, 1, easy\_Bar\_2);

            break;

        case 3:

            change\_Game\_Board(cor, 1, 2, easy\_Bar\_1);

            break;

        case 4:

            change\_Game\_Board(cor, 2, 2, easy\_Square);

            break;

        case 5:

            change\_Game\_Board(cor, 2, 2, Norm\_L\_1);

            break;

        case 6:

            change\_Game\_Board(cor, 2, 2, Norm\_L\_2);

            break;

        case 7:

            change\_Game\_Board(cor, 2, 2, Norm\_L\_3);

            break;

        case 8:

            change\_Game\_Board(cor, 2, 2, Norm\_L\_4);

            break;

        case 9:

            change\_Game\_Board(cor, 3, 1, Norm\_Bar\_3\_1);

            break;

        case 10:

            change\_Game\_Board(cor, 1, 3, Norm\_Bar\_3\_2);

            break;

        case 11:

            change\_Game\_Board(cor, 2, 3, Hard\_LL\_1);

            break;

        case 12:

            change\_Game\_Board(cor, 2, 3, Hard\_LL\_2);

            break;

        case 13:

            change\_Game\_Board(cor, 3, 2, Hard\_LL\_3);

            break;

        case 14:

            change\_Game\_Board(cor, 3, 2, Hard\_LL\_4);

            break;

        case 15:

            change\_Game\_Board(cor, 2, 3, Hard\_L\_1);

            break;

        case 16:

            change\_Game\_Board(cor, 2, 3, Hard\_L\_2);

            break;

        case 17:

            change\_Game\_Board(cor, 2, 3, Hard\_L\_3);

            break;

        case 18:

            change\_Game\_Board(cor, 2, 3, Hard\_L\_4);

            break;

        case 19:

            change\_Game\_Board(cor, 2, 3, Hard\_T\_1);

            break;

        case 20:

            change\_Game\_Board(cor, 2, 3, Hard\_T\_2);

            break;

        case 21:

            change\_Game\_Board(cor, 3, 2, Hard\_T\_3);

            break;

        case 22:

            change\_Game\_Board(cor, 3, 2, Hard\_T\_4);

            break;

        case 23:

            change\_Game\_Board(cor, 1, 4, Hard\_Bar\_4\_1);

            break;

        case 24:

            change\_Game\_Board(cor, 4, 1, Hard\_Bar\_4\_2);

            break;

        }

        cord = 0;

        return;

    }

کد تابع Do

وظیفه این تابع فرستادن مقادیر لازم به تابع change\_Game\_Board باتوجه به شماره شکل هست.

تابع Void Row\_Checker

    void Row\_Checker()

    {

        bool f[2];

        int y = 0;

        int x = 0;

        for (int i = 0; i < boardsize[1]; i++)

        {

            f[0] = true;

            for (int j = 0; j < boardsize[0]; j++)

            {

                if (!Board[j][i])

                {

                    f[0] = false;

                    break;

                }

            }

            if (f[0])

            {

                y = i;

                break;

            }

        }

        if (f[0])

        {

            for (int i = y; i > 0; i--)

            {

                for (int j = 0; j < boardsize[0]; j++)

                {

                    Board[j][i] = Board[j][i - 1];

                }

            }

            for (int j = 0; j < boardsize[0]; j++)

            {

                Board[j][0] = false;

            }

            Score += 10;

            return Row\_Checker();

        }

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

        {

            f[1] = true;

            for (int j = 0; j < boardsize[1]; j++)

            {

                if (!Board[i][j])

                {

                    f[1] = false;

                    break;

                }

            }

            if (f[1])

            {

                x = i;

                break;

            }

        }

        if (f[1])

        {

            for (int j = 0; j < boardsize[1]; j++)

            {

                Board[x][j] = false;

            }

            Score += 10;

            return Row\_Checker();

        }

    }

کد تابع Row\_Checker

این تابع اول چک میکند که ردیف افقی کامل شده ای وجود دارد یا نه که در صورت وجود آن ردیف را حذف و ردیف های بالای آنر را یک ردیف پایین می‌کشاند و تلبع را دوباره برای چک دوباره باز خوانی می‌کند و در صورت نبود چک می‌کند ردیف عمودی وجود دارد یا نه اگر وجود داشت آن را پاک کرده و تابع را بزا خوانی می‌کند وگرنه تابع متوقف می‌شود .

تابع Void Move

    bool Move(int cor, bool right, int x, bool obj[4][4])

    {

        if (right)

            cor++;

        else

            cor--;

        if (right && cor + x > boardsize[0])

            return false;

        if (!right && cor <= -1)

            return true;

        Copy\_Board\_To\_Tmp();

        for (int i = 0; i < x; i++)

        {

            for (int j = 0; j < 4; j++)

            {

                if (obj[i][j])

                {

                    if (Tmp[i + cor][j])

                        return Move(cor, right, x, obj);

                    Tmp[i + cor][j] = true;

                }

            }

        }

        cord = cor;

        Render\_Game\_Board(Tmp);

        return true;

    }

کد تابع Move

این تابع وظیفه جابجایی شیئ در حال حرکت بر روی صفحه بازی را دارد و اگر شیئ در کل صفحه امکان انداخن نداشته باشد مقدار false را خروجی میدهد وگرنه اگر امکان ادناخت بر مانی را نداشت ولی مکان بعدی را داشت در مکان بعدی آن را نمایش می‌دهد و اگذ ورودی Right درست باشد یعنی این تابع باید شیئ را به سمت راست برده وگرنه به شمت چپ ببرد.

تابع Void Move\_mod

    bool Move\_mod(int mod, bool R)

    {

        switch (mod)

        {

        case 1:

            return Move(cord, R, 1, easy\_Single);

            break;

        case 2:

            return Move(cord, R, 2, easy\_Bar\_2);

            break;

        case 3:

            return Move(cord, R, 1, easy\_Bar\_1);

            break;

        case 4:

            return Move(cord, R, 2, easy\_Square);

            break;

        case 5:

            return Move(cord, R, 2, Norm\_L\_1);

            break;

        case 6:

            return Move(cord, R, 2, Norm\_L\_2);

            break;

        case 7:

            return Move(cord, R, 2, Norm\_L\_3);

            break;

        case 8:

            return Move(cord, R, 2, Norm\_L\_4);

            break;

        case 9:

            return Move(cord, R, 3, Norm\_Bar\_3\_1);

            break;

        case 10:

            return Move(cord, R, 1, Norm\_Bar\_3\_2);

            break;

        case 11:

            return Move(cord, R, 2, Hard\_LL\_1);

            break;

        case 12:

            return Move(cord, R, 2, Hard\_LL\_2);

            break;

        case 13:

            return Move(cord, R, 3, Hard\_LL\_3);

            break;

        case 14:

            return Move(cord, R, 3, Hard\_LL\_4);

            break;

        case 15:

            return Move(cord, R, 2, Hard\_L\_1);

            break;

        case 16:

            return Move(cord, R, 2, Hard\_L\_2);

            break;

        case 17:

            return Move(cord, R, 2, Hard\_L\_3);

            break;

        case 18:

            return Move(cord, R, 2, Hard\_L\_4);

            break;

        case 19:

            return Move(cord, R, 2, Hard\_T\_1);

            break;

        case 20:

            return Move(cord, R, 2, Hard\_T\_2);

            break;

        case 21:

            return Move(cord, R, 3, Hard\_T\_3);

            break;

        case 22:

            return Move(cord, R, 3, Hard\_T\_4);

            break;

        case 23:

            return Move(cord, R, 1, Hard\_Bar\_4\_1);

            break;

        case 24:

            return Move(cord, R, 4, Hard\_Bar\_4\_2);

            break;

        }

        return false;

    }

کد تابع Move\_mod

کار این تابع دادن مقادیر مرود نیاز به تابع Move با توجه به شکل و ابعد جسم هست

تابع Void Next\_Block

    void Next\_Block(int x, int y, bool obj[4][4])

    {

        for (int i = 0; i < 4; i++)

        {

            ConXY(x, y + i);

            for (int j = 0; j < 4; j++)

            {

                if (obj[j][i])

                    cout << '#';

                else

                    cout << ' ';

            }

        }

    }

کد تابع Next\_Block

وظیفه این تابع چاپ بلوک داده شده در کنسول در جای مشخص شده در کنسول می‌باشد.

تابع Void Next\_mod

    void Next\_mod(int x, int y, int mod)

    {

        switch (mod)

        {

        case 1:

            return Next\_Block(x, y, easy\_Single);

            break;

        case 2:

            return Next\_Block(x, y, easy\_Bar\_2);

            break;

        case 3:

            return Next\_Block(x, y, easy\_Bar\_1);

            break;

        case 4:

            return Next\_Block(x, y, easy\_Square);

            break;

        case 5:

            return Next\_Block(x, y, Norm\_L\_1);

            break;

        case 6:

            return Next\_Block(x, y, Norm\_L\_2);

            break;

        case 7:

            return Next\_Block(x, y, Norm\_L\_3);

            break;

        case 8:

            return Next\_Block(x, y, Norm\_L\_4);

            break;

        case 9:

            return Next\_Block(x, y, Norm\_Bar\_3\_1);

            break;

        case 10:

            return Next\_Block(x, y, Norm\_Bar\_3\_2);

            break;

        case 11:

            return Next\_Block(x, y, Hard\_LL\_1);

            break;

        case 12:

            return Next\_Block(x, y, Hard\_LL\_2);

            break;

        case 13:

            return Next\_Block(x, y, Hard\_LL\_3);

            break;

        case 14:

            return Next\_Block(x, y, Hard\_LL\_4);

            break;

        case 15:

            return Next\_Block(x, y, Hard\_L\_1);

            break;

        case 16:

            return Next\_Block(x, y, Hard\_L\_2);

            break;

        case 17:

            return Next\_Block(x, y, Hard\_L\_3);

            break;

        case 18:

            return Next\_Block(x, y, Hard\_L\_4);

            break;

        case 19:

            return Next\_Block(x, y, Hard\_T\_1);

            break;

        case 20:

            return Next\_Block(x, y, Hard\_T\_2);

            break;

        case 21:

            return Next\_Block(x, y, Hard\_T\_3);

            break;

        case 22:

            return Next\_Block(x, y, Hard\_T\_4);

            break;

        case 23:

            return Next\_Block(x, y, Hard\_Bar\_4\_1);

            break;

        case 24:

            return Next\_Block(x, y, Hard\_Bar\_4\_2);

            break;

        }

    }

کد تابع Next\_mod

وظیفه این تابع ارسال مقادیر لازم به تابع Next\_Block با توجه به شماره شکل می‌باشد.

توابع عمومی

تابع Void start\_menu

    void start\_menu()

    {

        system("cls");

        cout << "oooooooooo o888                        oooo         oooooooooo                                      o888             \n 888    888 888   ooooooo     ooooooo   888  ooooo   888    888 oooo  oooo  ooooooooooo ooooooooooo  888  ooooooooo8 \n 888oooo88  888 888     888 888     888 888o888      888oooo88   888   888       8888        8888    888 888oooooo8  \n 888    888 888 888     888 888         8888 88o     888         888   888    8888        8888       888 888         \no888ooo888 o888o  88ooo88     88ooo888 o888o o888o  o888o         888o88 8o o888ooooooo o888ooooooo o888o  88oooo888 \n ";

        int set[3] = {116, 7, 7};

        int X = 50;

        int Y = 10;

        int tmp = 0;

        ConClr(7);

        ConXY(X - 16, Y + 15);

        cout << "  Use up & down Arrow keys for menu navigating ";

        ConXY(X - 16, Y + 16);

        cout << "Telegram : @JVUXE | @TheRealStan\_313 | @mmrezaw";

        int counter = 1;

        char key;

        char keyz;

        while (true)

        {

            ConXY(X + 6, Y);

            ConClr(set[0]);

            cout << "START";

            ConXY(X, Y + 1);

            ConClr(set[1]);

            cout << "Select Difficulty";

            ConXY(X + 6, Y + 2);

            ConClr(set[2]);

            cout << "Exit";

            ConClr(7);

            key = \_getch();

            if (key == 13)

            {

                if (counter == 1)

                {

                    system("cls");

                    return;

                }

                if (counter == 2)

                {

                    set[0] = 116;

                    set[1] = 7;

                    set[2] = 7;

                    X = 20;

                    Y = 15;

                    counter = 1;

                    ConXY(X, Y - 1);

                    ConClr(7);

                    cout << "Select Difficulty";

                    while (true)

                    {

                        ConXY(X + 5, Y);

                        ConClr(set[0]);

                        cout << "Defualt";

                        ConXY(X + 5, Y + 1);

                        ConClr(set[1]);

                        cout << "Custom";

                        ConXY(X + 6, Y + 2);

                        ConClr(set[2]);

                        cout << "Back";

                        ConClr(7);

                        key = \_getch();

                        if (key == 13)

                        {

                            if (counter == 1)

                            {

                                diff[0] = 1;

                                diff[1] = 1;

                                diff[2] = 1;

                                boardsize[0] = 11;

                                boardsize[1] = 11;

                                system("cls");

                                return start\_menu();

                            }

                            if (counter == 2)

                            {

                                ConXY(X + 20, Y - 1);

                                ConClr(207);

                                cout << "INPUT VALUES OF LENGTHS MUST BIGGER THEN 11";

                                ConClr(228);

                                ConXY(X + 20, Y);

                                cout << "Input vertical length";

                                ConClr(0);

                                cin >> tmp;

                                if (tmp >= 11)

                                {

                                    ConClr(7);

                                    ConXY(X + 20, Y);

                                    cout << "                                                 ";

                                    boardsize[1] = tmp;

                                    ConClr(228);

                                    ConXY(X + 20, Y);

                                    cout << "Input Horizental length";

                                    ConClr(0);

                                    cin >> tmp;

                                    if (tmp >= 11)

                                    {

                                        boardsize[0] = tmp;

                                        ConClr(7);

                                        ConXY(X + 20, Y);

                                        cout << "                                                 ";

                                        ConXY(X + 20, Y - 1);

                                        cout << "                                                ";

                                        set[0] = 116;

                                        set[1] = 7;

                                        set[2] = 7;

                                        X = 50;

                                        Y = 15;

                                        counter = 1;

                                        ConXY(X, Y - 1);

                                        ConClr(7);

                                        cout << "Select Difficulty";

                                        while (true)

                                        {

                                            ConXY(X + 6, Y);

                                            ConClr(set[0]);

                                            cout << "HARD";

                                            ConXY(X + 5, Y + 1);

                                            ConClr(set[1]);

                                            cout << "NORMALL";

                                            ConXY(X + 6, Y + 2);

                                            ConClr(set[2]);

                                            cout << "EASY";

                                            ConClr(7);

                                            key = \_getch();

                                            if (key == 13)

                                            {

                                                if (counter == 1)

                                                {

                                                    diff[0] = 3;

                                                    diff[1] = 2;

                                                    diff[2] = 1;

                                                    return start\_menu();

                                                }

                                                if (counter == 2)

                                                {

                                                    diff[0] = 1;

                                                    diff[1] = 1;

                                                    diff[2] = 1;

                                                    return start\_menu();

                                                }

                                                if (counter == 3)

                                                {

                                                    diff[0] = 1;

                                                    diff[1] = 2;

                                                    diff[2] = 3;

                                                    return start\_menu();

                                                }

                                            }

                                            if (key == -32)

                                            {

                                                keyz = \_getch();

                                                if (keyz == 72 && counter != 1)

                                                    counter--;

                                                if (keyz == 80 && counter != 3)

                                                    counter++;

                                            }

                                            set[0] = 7;

                                            set[1] = 7;

                                            set[2] = 7;

                                            set[counter - 1] = 116;

                                        }

                                    }

                                    ConClr(79);

                                    ConXY(X + 20, Y);

                                    cout << "INPUT VALUE IS INVALID  PRESS ANY KEY TO BACK";

                                    \_getch();

                                    ConClr(7);

                                    ConXY(X + 20, Y);

                                    cout << "                                                 ";

                                    ConXY(X + 20, Y - 1);

                                    cout << "                                                ";

                                }

                            }

                            if (counter == 3)

                            {

                                system("cls");

                                return start\_menu();

                            }

                        }

                        if (key == -32)

                        {

                            keyz = \_getch();

                            if (keyz == 72 && counter != 1)

                                counter--;

                            if (keyz == 80 && counter != 3)

                                counter++;

                        }

                        set[0] = 7;

                        set[1] = 7;

                        set[2] = 7;

                        set[counter - 1] = 116;

                    }

                }

                if (counter == 3)

                {

                    system("cls");

                    exit(0);

                }

            }

            if (key == -32)

            {

                keyz = \_getch();

                if (keyz == 72 && counter != 1)

                    counter--;

                if (keyz == 80 && counter != 3)

                    counter++;

            }

            set[0] = 7;

            set[1] = 7;

            set[2] = 7;

            set[counter - 1] = 116;

        }

    }

کد تابع start\_menu

وظیفه این تابع ساخت منو شروع بازی هست که با توجه به ورودی های کاربر اگر دکمه شروع بازی را انتخاب کرده از منو شروع خارج می‌شود.اگر خروج را انتخاب کرده کامل از برنامه خارج می‌شود. و لاگر منو انتخاب درجه سختی را انتخاب کرده با سه گزینه بازگشت به منو شروع یازی ، پیش فرض بازی یا گیزنه کاستوم که در صورت انتخاب آن اول از کاربر طول و عرض صفحه بازی پرسیده می‌شود و سپس سختی یا آسونی بازی از کاربر پرسیده می‌شود.

تابع Void Play

    void Play()

    {

        system("cls");

        Make\_Game\_Borad();

        Score = 0;

        srand(time(0));

        int set[4] = {116, 7, 7, 0};

        int mod = RndSel();

        int nmod = RndSel();

        int X = 6 + 5 \* boardsize[0] / 2;

        int Y = -1 + boardsize[1] / 2;

        ConClr(95);

        ConXY(X - 6, Y - 1);

        cout << "Next";

        Next\_mod(X - 6, Y, nmod);

        ConClr(7);

        cord = -1;

        Move\_mod(mod, true);

        int counter = 1;

        char key;

        char keyz;

        bool high = false;

        while (true)

        {

            if (Score > HighScore)

            {

                if (!high)

                {

                    High\_Score(X, Y);

                    high = true;

                }

                HighScore = Score;

            }

            ConXY(X, Y);

            ConClr(set[0]);

            cout << "Retry";

            ConXY(X, Y + 1);

            ConClr(set[1]);

            cout << "Hint";

            ConXY(X, Y + 2);

            ConClr(set[2]);

            cout << "Exit";

            ConXY(X, Y + 3);

            ConClr(set[3]);

            cout << "Clear a row";

            ConClr(7);

            ConXY(X, Y + 4);

            cout << "High SCORE : " << HighScore;

            ConXY(X, Y + 5);

            cout << "SCORE : " << Score;

            key = \_getch();

            if (key == 13)

            {

                if (counter == 1)

                {

                    if (HighScore < Score)

                        HighScore = Score;

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

                    {

                        for (int j = 0; j < boardsize[1]; j++)

                            Board[i][j] = false;

                    }

                    return Play();

                }

                if (counter == 2)

                {

                    Hint(X, Y);

                }

                if (counter == 3)

                {

                    Exit();

                }

                if (counter == 4)

                {

                    Score -= 50;

                    Clear\_a\_Row();

                    if (Score < 50)

                        counter--;

                }

            }

            if (key == -32)

            {

                keyz = \_getch();

                if (keyz == 72 && counter > 1)

                    counter--;

                if (keyz == 80 && ((Score >= 50 && counter < 4) || (counter < 3)))

                {

                    counter++;

                }

            }

            if (key == 'a' || key == 'A')

                Move\_mod(mod, false);

            if (key == 'd' || key == 'd')

                Move\_mod(mod, true);

            if (key == 's' || key == 'S')

            {

                Do(cord, mod);

                Row\_Checker();

                cord = -1;

                mod = nmod;

                nmod = RndSel();

                ConClr(95);

                Next\_mod(X - 6, Y, nmod);

                ConClr(7);

                if (!Move\_mod(mod, true))

                {

                    if (HighScore < Score)

                        HighScore = Score;

                    Game\_Over();

                    break;

                }

            }

            set[0] = 7;

            set[1] = 7;

            set[2] = 7;

            if (Score >= 50)

                set[3] = 43;

            else

                set[3] = 0;

            set[counter - 1] = 116;

        }

        return Play();

    }

کد تابع Play

این تابع وظیفه کنترل بازی را دارد که یک صفحه بازی دو منو کنار بازی سه درصورت رکورد فرد چاپ رکورد پیام چهار درصورت انتخاب گزینه Hint نمایش راهنمای کنترل بازی به کاربر پنج در صورت باخت نمایش پیام Game Over به کاربر.

تابع Main

int main()

{

    Game game;

    game.start\_menu();

    game.Play();

    return 0;

}

کد تابع Main