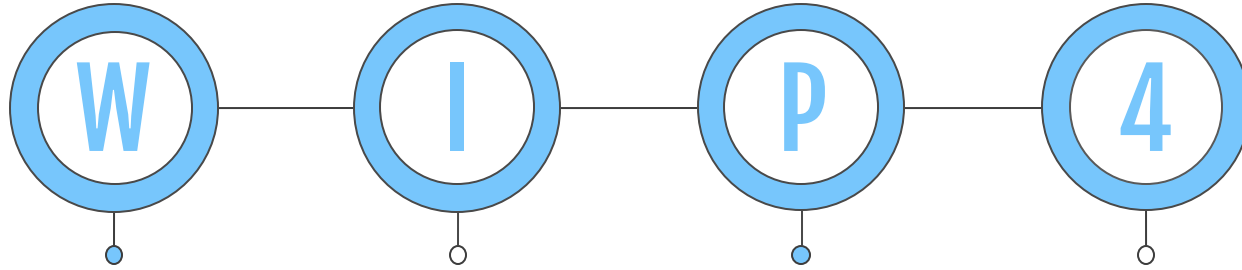


WIP 4 Work in Progress

Gioca a Forza 4 con i tuoi
amici



Il Progetto



Work

Il motto del team di sviluppo è
"Lavorando si impara"

In

In qualsiasi luogo si lavora, anche sui mezzi pubblici

Progress

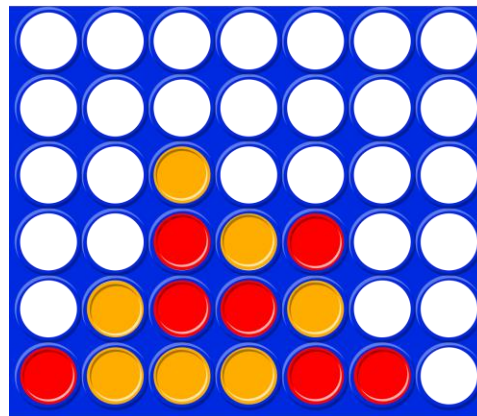
Ogni giorno sempre un passo avanti

Forza 4

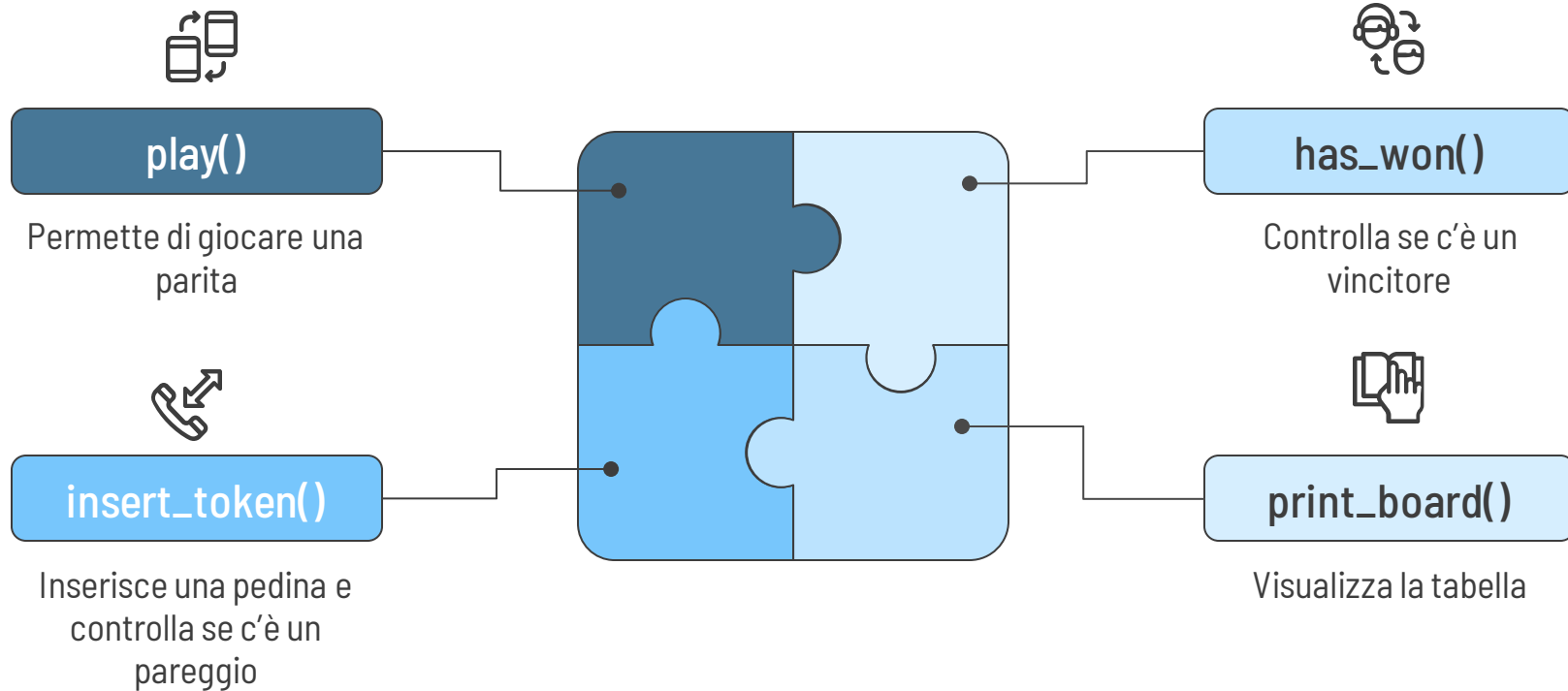
Lo scopo del progetto è realizzare il famoso gioco di Forza 4

Il Gioco

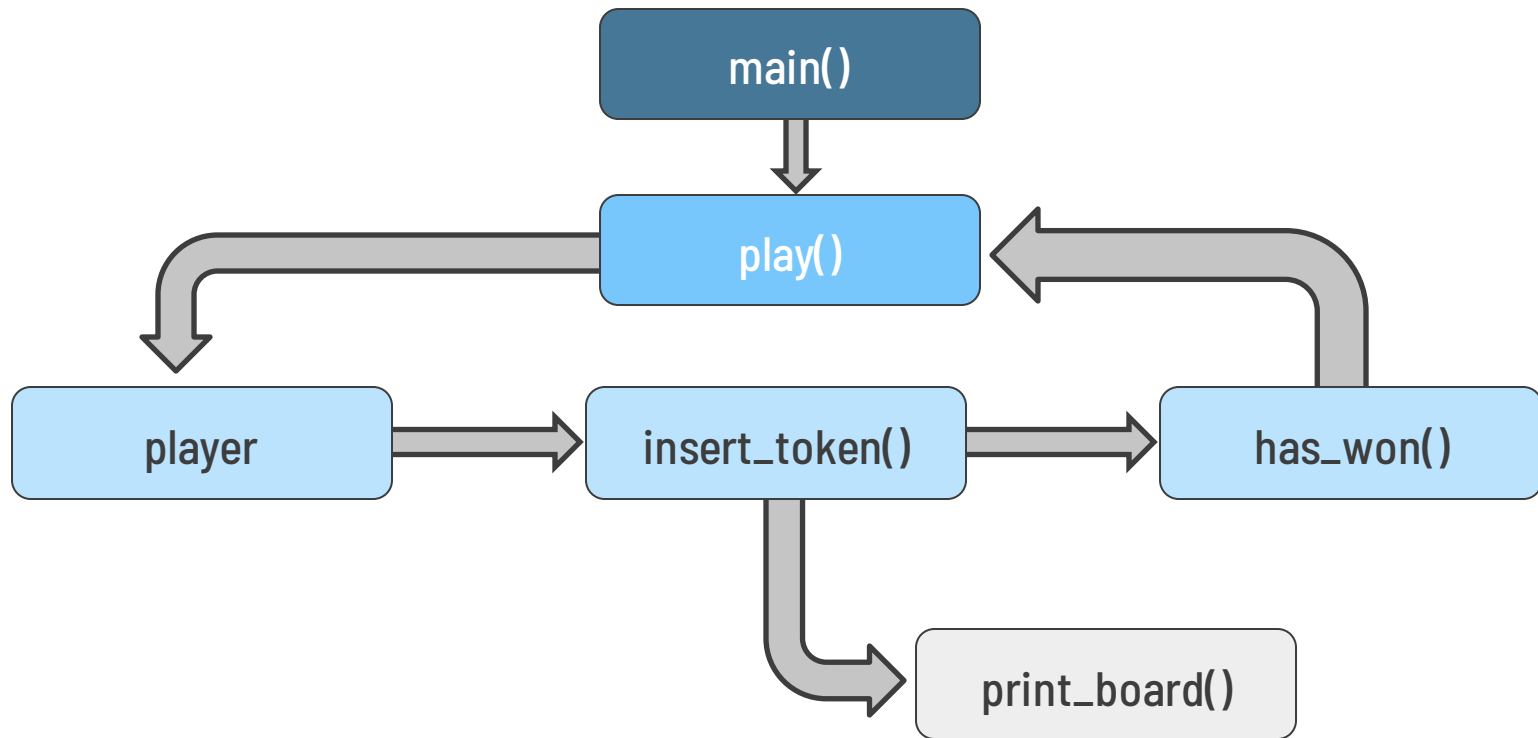
- Composto da una griglia 6x7 e da pedine da inserire
- Massimo 2 giocatori
- Allinea 4 pedine in orizzontale, verticale o diagonale per vincere
- In possesso della MB (Milton Bradley) dal 1984

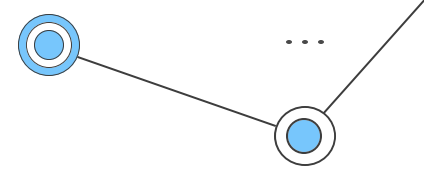
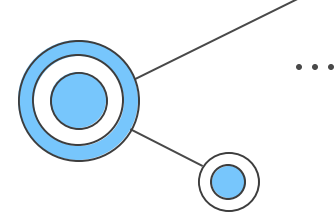


Suddivisione del Programma



Flusso de Codice





{ Analisi del Codice }

MAIN()

```
int main(){
    setlocale(LC_ALL, ""); // Per usare i caratteri unicode

    int board[RIGHE][COLONNE] = {0};
    int scores[PLAYERS] = {0};
    int last_winner = 2;
    int win_row[4][2];

    clear();
    print_bold("Benvenuto su WIP4\nGioca a forza 4 direttamente dal terminale.\n");
    printf("Premi un tasto per iniziare una partita o 'q' per uscire...\n");
    icanon_mode(false);

    if(getchar() != 'q'){
        do{

            icanon_mode(true);
            play(board, scores, %26last_winner, win_row);
            print_bold("Punteggi:\n");

            for(int i = 0; i < PLAYERS; i%2B%2B)
                printf("Giocatore %d: %d\n", i %2B 1, scores[i]);

            printf("Premi un tasto per giocare ancora o 'q' per uscire\n");

            icanon_mode(false);

            reset_matrix(RIGHE, COLONNE, board);

        }while(getchar() != 'q');

        print_bold("\b \b\nBruciassero gli Unicode\n");

    } else
        printf("\b \b");

    return 0;
}
```

La funzione principale contiene le inizializzazioni delle variabili e il ciclo per giocare a forza 4.



PLAY()

```
void play(int board[RIGHE][COLONNE], int scores[], int *last_winner, int win_row[4][2]){
    bool not_won = true;
    int player = *last_winner;
    bool tie;

    do{
        int x, y;

        player = rotate_number(1, 2, player, 1);

        insert_token(board, player, &x, &y, &tie);

        clear();

        if(tie){
            printf("Pareggio!\n");
            break;
        }

        if(has_won(board, player, x, y, win_row)){
            not_won = false;
            scores[player-1]++;
        }

    } while(not_won);

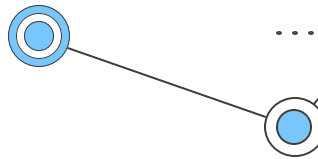
    *last_winner = player;
    print_board(board, win_row, !not_won);
    printf("Ha vinto il giocatore %d!\n", player);
}
```

Gestore delle singole partite.

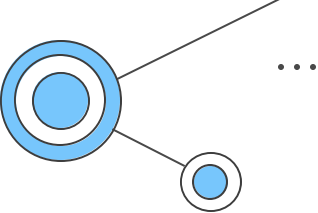


INSERT_TOKEN()

```
void insert_token(int board[RIGHE][COLONNE], int player, int *x, int *y, bool *tie){
    char c;
    *y = 0;
    int go_to_val = 0;
    *tie = false;
    int t[4][2];
    clear();
    icanon_mode(false);
    do{ // Loop finchè non trova una colonna non piena
        do{ // Loop finchè l'utente non preme invio
            // Controlla se le prime colonne sono piene e sposta il cursore
            int arrow_pos = 0;
            for(int i = 0; i < COLONNE; i++){
                if(board[0][i] == 0)
                    break;
                arrow_pos++;
            }
            if(arrow_pos > go_to_val && *y < COLONNE-1){
                (*y) = arrow_pos;
                go_to(4+(2*(arrow_pos-1)), 0);
                go_to_val = 4+(2*(arrow_pos-1));
            }
            if(arrow_pos == COLONNE-1){
                *tie = true;
                return;
            }
            printf("%ls\n", DOWN_ARROW);
            print_board(board, t, false);
            printf("turno del giocatore %d:\n", player);
            c = getchar();
            if(c == 'C'){
                if(*y < COLONNE-1){
                    (*y)++;
                    go_to_val += (go_to_val==0) ? 4 : 2;
                }
            }
            if(c == 'D'){
                if(*y > 0){
                    (*y)--;
                    go_to_val -= (go_to_val==4) ? 4 : 2;
                }
            }
            clear();
            go_to(go_to_val, 0);
        }while(c != '\n');
    }while(board[0][*y] != 0);
    icanon_mode(true);
    for(int i = RIGHE - 1; i >= 0; i--){
        if(board[i][*y] == 0){
            board[i][*y] = player;
            *x = i;
            break;
        }
    }
}
```

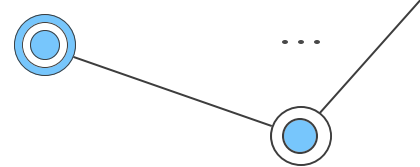


Inserimento di una pedina all'interno della griglia mediante un cursore. Controlla un eventuale pareggio.



HAS_WON()

```
bool has_won(int board[RIGHE][COLONNE], int player, int x, int y, int win_row[4][2]){
    for(int cast = 0; cast < 4; cast++){
        bool temp_won;
        int increasing_x, increasig_y;
        int won_count = 1;
        int temp_x = x;
        int temp_y = y;
        reset_matrix(4, 2, win_row);
        win_row[0][0] = temp_x;
        win_row[0][1] = temp_y;
        do{
            switch(cast){
                case 0: // orizzontali
                    increasing_x = 0;
                    increasig_y = -1;
                    break;
                case 1: // verticali
                    increasing_x = -1;
                    increasig_y = 0;
                    break;
                case 2: // diagonale
                    increasing_x = -1;
                    increasig_y = 1;
                    break;
                case 3: // diagonale inversa
                    increasing_x = -1;
                    increasig_y = -1;
                    break;
            }
            if(temp_x < 0 || temp_x >= RIGHE || temp_y < 0 || temp_y >= COLONNE)
                temp_won = false;
            else
                temp_won = (board[temp_x + increasing_x][temp_y + increasig_y] == player) && (temp_x +
                increasing_x < RIGHE && temp_x + increasing_x >= 0);
            if(temp_won){
                temp_x += increasing_x;
                temp_y += increasig_y;
                win_row[won_count][0] = temp_x;
                win_row[won_count][1] = temp_y;
                won_count++;
            } else{
                temp_y = y;
                temp_x = x;
                bool temp_won2;
                increasing_x *= -1;
                increasig_y *= -1;
                do{
                    if(temp_x < 0 || temp_x >= RIGHE || temp_y < 0 || temp_y >= COLONNE)
                        temp_won2 = false;
                    else
                        temp_won2 = board[temp_x + increasing_x][temp_y + increasig_y] == player &&
                        (temp_x + increasing_x < RIGHE && temp_x + increasing_x >= 0);
                    if(temp_won2){
                        temp_x += increasing_x;
                        temp_y += increasig_y;
                        win_row[won_count][0] = temp_x;
                        win_row[won_count][1] = temp_y;
                        won_count++;
                    } else {
                        continue;
                    }
                }while(temp_won2);
            }
        }while(temp_won);
        if(won_count >= 4)
            return true;
        else
            return false;
    }
}
```



Controllo di un'eventuale vincita di un giocatore data la sua ultima mossa.

PRINT_BOARD()

```
void print_board(int board[RIGHE][COLONNE], int win_row[4][2], bool won){  
  
    printf("%ls", LINE_DR);  
    for(int i = 0; i < RIGHE; i++){  
        printf("%ls", LINE_HOR LINE_INCROCIO);  
        printf("%ls", LINE_HOR LINE_DL);  
        for(int j = 0; j < RIGHE-1; j++){  
            printf("\n%ls", LINE_VER);  
            for(int i = 0; i < COLONNE; i++){  
                printf("%c", find_value(board, j, i, win_row, won));  
                set_color(STD_COLOR);  
                printf("%ls", LINE_VER);  
            }  
            printf("\n%ls", LINE_BORDOL);  
            for(int i = 0; i < RIGHE; i++){  
                printf("%ls", LINE_HOR LINE_CROCE);  
                printf("%ls", LINE_HOR LINE_BORDOR);  
            }  
            printf("\n%ls", LINE_VER);  
            for(int i = 0; i < COLONNE; i++){  
                printf("%c", find_value(board, RIGHE-1, i, win_row, won));  
                set_color(STD_COLOR);  
                printf("%ls", LINE_VER);  
            }  
            printf("\n%ls", LINE_UR);  
            for(int i = 0; i < RIGHE; i++){  
                printf("%ls", LINE_HOR LINE_INCROCIOS);  
                printf("%ls\n", LINE_HOR LINE_UL);  
            }  
        }  
    }
```

```
char find_value(int board[RIGHE][COLONNE], int righe, int colonne, int win_row[4][2], bool won){  
    bool evidenza = false;  
    if(won)  
        for(int i = 0; i < 4; i++){  
            if(righe == win_row[i][0] && colonne == win_row[i][1])  
                evidenza = true;  
        }  
    switch(board[righe][colonne]){  
        case 0:  
            return ' ';  
            break;  
        case 1:  
            set_color(evidenza ? PLAYER1_WIN : PLAYER1_COLOR);  
            return 'O';  
            break;  
        case 2:  
            set_color(evidenza ? PLAYER2_WIN : PLAYER2_COLOR);  
            return 'X';  
            break;  
        default:  
            return '\0';  
    }  
}
```

print_board(): stampa la griglia.

find_value(): restituisce il carattere da stampare e imposta il suo colore.

Funzioni secondarie

```
void icanon_mode(bool enable){
    struct termios term_settings;

    // Prende le impostazioni del terminale
    tcgetattr(STDIN_FILENO, &term_settings);

    if(enable){

        // Abilita modalità ICANON
        term_settings.c_lflag |= ICANON;

    } else{

        // Toglie modalità ICANON
        term_settings.c_lflag &= ~ICANON;

        // Imposta il minimo numero di caratteri
        term_settings.c_cc[VMIN] = 1;

        // Imposta il timeout in secondi
        term_settings.c_cc[VTIME] = 0;

    }

    // Applica impostazioni
    tcsetattr(STDIN_FILENO, TCSANOW, &term_settings);
}
```

```
int rotate_number(int start, int end, int numero, int quanto_ruota){
    int range = end - start + 1;
    return start + ((numero - start + quanto_ruota) % range + range) % range;
}
```

icanon_mode(): imposta la modalità ICANON dato come parametro true e vice versa.

rotate_number(): ruota un numero dato il suo range d'azione.



Problemi Riscontrati

UNICODE

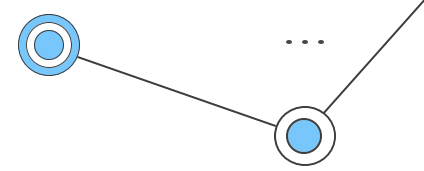
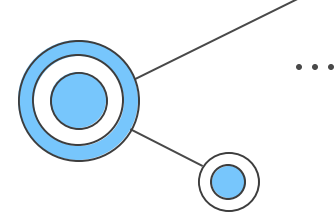
La griglia, composta da Unicode, è stata una delle *challenge* più difficili da realizzare graficamente.

CURSORE

Ci sono molte strategie per far muovere un cursore, il team WIP4 ha realizzato quella più *user friendly*.

VINCITA

Abbiamo studiato il modo più efficiente per controllare tutti i casi di vittoria.



Grazie per l'Attenzione
Il Team WIP4