plik snake.h to plik nagłówkowy i zawiera wszystkie definicje globalnych zmiennych używanych w programie głównym. Pełni on funkcje konfiguracji działania programu. Przykładem może być definicja kolorów wyświetlanych na ekranie.

```
/* kolor gry */
#define SNAKE_COLOR_BORDER      ILI9341_COLOR_BLACK
#define SNAKE_COLOR_MAIN_BCK    ILI9341_COLOR_BLUE
#define SNAKE_COLOR_GAME_BCK    ILI9341_COLOR_GREEN
#define SNAKE_COLOR_PIXEL       ILI9341_COLOR_BROWN
#define SNAKE_COLOR_TARGET      ILI9341_COLOR_BLACK
#define SNAKE_COLOR_HEAD        ILI9341_COLOR_RED
```

Poniższy kod przedstawia bindowanie komend wysyłanych z klawiatury z zmiennymi które odczytuje się w grze w celu sprawdzenia wejść z klawiatury i potencjalnych ruchów.

```
#define SNAKE_KEY_LEFT          ((uint8_t) 'a') // 'a'
#define SNAKE_KEY_RIGHT         ((uint8_t) 'd') // 'd'
#define SNAKE_KEY_UP            ((uint8_t) 'w') // 'w'
#define SNAKE_KEY_DOWN          ((uint8_t) 's') // 's'
#define SNAKE_KEY_SPEED_UP      ((uint8_t) 'u') // 'u'
#define SNAKE_KEY_SPEED_DOWN    ((uint8_t) 'i') // 'i'
#define SNAKE_KEY_PAUSE         ((uint8_t) 'p') // 'p'
#define SNAKE_KEY_RESET         ((uint8_t) 'r') // 'r'
#define SNAKE_KEY_OVERFLOW      ((uint8_t) 'm') // 'm'
```

Poniższa struktura tworzy “obiekt” węża składającego się z jego długości, ostatniego indeksu, kierunku i wyniku.

```
typedef struct {
   int8_t Snake[SNAKE_PIXELS * SNAKE_PIXELS][2];
   uint16_t Length;
   uint16_t LastIndex;
   uint8_t Direction;
   uint16_t Hits;
} SNAKE_t;
```

Poniższa struktura jest używana jako struktura odpowiedzialną z akonfigurację rozgrywki, pole overflow odpowiada za tryb gry czy wąż może przenikać przez ściany czy nie. Zmienna speed przechowuje obecną prędkość, micros odpowiada a przeko w cyklu, pause jest równe True jeśli włączymy pause gry.

```
typedef struct {
   uint8_t Overflow;
   uint8_t Speed;
   uint32_t Micros;
   uint8_t Pause;
} SNAKE_Settings_t;
```

Inicjacja biblioteki obsługi klawiatury i stworzenie jej obiektu

```
USB_HIDHOST_Keyboard_t Keyboard;
```

Owa funkcja to główna pętla programu odpowiedzialna początkową inicjację programu oraz późniejszą obsługę stanów i zachowania gry.

```
void SNAKE_Start(void)
```

Poniższy kod przedstawia inicjacje systemowych bibliotek random, delay, wyświetlacza lcd, hosta usb, oraz struktur z pliku nagłówkowego.

```
// inicjalizacja delaya
   DELAY_Init();
   //inicjalizacja randoma
   RNG_Init();
   //inicjalizacja LCD
   ILI9341_Init();
   ILI9341_Rotate(TM_ILI9341_Orientation_Portrait_2);
   //inicjalizacja hosta USB
   USB_HIDHOST_Init();
   //ustawianie dafaultowych ustawień
   SNAKE_SetFirstOptions();
   //ustawianie dafaultowych wartości
   SNAKE_SetDefaultSnake();
   //przygotowanie wyświetlacza
   SNAKE_PrepareDisplay();
   //ustawienie targetu generatoa random
   SNAKE_GenerateTarget();
   //ustawienie czasu na 0
   DELAY_SetTime(0);
```

Poniższy if obsługuje wejście klawiatury i w przypadku wystąpienia zbudowanego stanu wykonuje skręcenie wężem

```
if (!Snake_FirstTime) {
               //przemieszczenie snake'a
               switch (Snake.Direction) {
                   case SNAKE_DIRECTION_LEFT:
                       Snake_Head[0] -= 1;
                       break;
                   case SNAKE_DIRECTION_RIGHT:
                       Snake_Head[0] += 1;
                       break;
                   case SNAKE_DIRECTION_UP:
                       Snake_Head[1] -= 1;
                       break;
```

```
                case SNAKE_DIRECTION_DOWN:
                    Snake_Head[1] += 1;
                    break;
                default:
                    break;
            }
        }
```

Poniższy kod obsługuję 2 tryby gry gdzie przenikanie przez ścianę jest dozwolone i drugi gdzie w takim przypadku wystąpi koniec gry

```
//overflow = tryb przechodzenia przez ścianę
        if (Settings.Overflow) {
            //sprawdzenie wartości X
            if (Snake_Head[0] == -1) {
                Snake_Head[0] = SNAKE_PIXELS - 1;
            } else if (Snake_Head[0] == SNAKE_PIXELS) {
                Snake_Head[0] = 0;
            }
            //sprawdzenie wartości Y
            if (Snake_Head[1] == -1) {
                Snake_Head[1] = SNAKE_PIXELS - 1;
            } else if (Snake_Head[1] == SNAKE_PIXELS) {
                Snake_Head[1] = 0;
            }
        } else {
            //sprawdzenie ścian kolizji
            if (
                Snake_Head[0] == -1 ||
                Snake_Head[0] == SNAKE_PIXELS ||
                Snake_Head[1] == -1 ||
                Snake_Head[1] == SNAKE_PIXELS
            ) {
                //koniec gry w przypadku uderzenia
                GameOver = 1;
            }
```

Dana instrukcja warunkowa wykonuje się raz podczas pierwszego cyklu pętli i ustawia początkowe parametry dla węża.

```
if (!Snake_FirstTime) {
            //usuwanie 1 wartosći z tablicy pozycji(końca ogona)
            SNAKE_DeleteFromArray(0, Snake_Foot);

            //sprawdzenie kolizji węża ze sobą
            if (SNAKE_MatchesSnakeLocations(Snake_Head)) {
```

```
                //koniec gry
                GameOver = 1;
            }
        }
```

Poniższy kod sprawdza czy wąż wjechał sam w siebie (game over) czy zjadł cukierka w tym przypadku następuje wygenerowanie nowego celu oraz inkrementacja wyniku

```
//sprawdzenie czy osiągnięto zadaną pozycję
        if (
            !GameOver &&
            Snake_Head[0] == Snake_Food[0] &&
            Snake_Head[1] == Snake_Food[1]
        ) {
            //dodanie nowego segmentu węża
            SNAKE_AddToArray(Snake_Head);
            // inkrementacja wyniku
            Snake.Hits++;
            //generowanie nowego cukierka
            SNAKE_GenerateTarget();
        }
```

Jeśli klawiatura jest podpięta czyli wynik tego ifa jest równy True program przechodzi do części odczytu danych z klawiatury.

```
                                if     (TM_USB_HIDHOST_Device()     ==
TM_USB_HIDHOST_Result_KeyboardConnected) {
```

W obsłudze klawiatury używane są zbindowane zmienne z snake.h jako case i w przypadku wykrycia danego przycisku wykonywane są poda canego case

```
    case SNAKE_KEY_LEFT:
                case SNAKE_KEY_RIGHT:
                case SNAKE_KEY_UP:
                case SNAKE_KEY_DOWN:
                case SNAKE_KEY_SPEED_UP:
                case SNAKE_KEY_SPEED_DOWN:
                case SNAKE_KEY_PAUSE:
                case SNAKE_KEY_RESET:
                case SNAKE_KEY_OVERFLOW:
```

Przykładowa obsługa skrętu w (lewo) podobnie jest w przypadku innych skrętów na początku jest sprawdzenie czy można wykonać skręt oraz potem następuje przypisanie kierunku do struktury węża oraz zerowanie pauzy

```
//zmiana kierunku jeśli jest możliwa
                        if (
                            Snake.Direction == SNAKE_DIRECTION_UP ||
                            Snake.Direction == SNAKE_DIRECTION_DOWN ||
                            Snake.Direction == SNAKE_DIRECTION_LEFT
                        ) {
                            //pauza gry
                            Settings.Pause = 0;
                            //ustawienie kierunku
                            Snake1.Direction = SNAKE_DIRECTION_LEFT;
                        }
```

Obsługa resetu węża:
-rysowanie od nowa planszy
-ustawianie ustawień defaultowych węża
-wygenerowanie nowego cukierka
-ustawienie stanu gameover
ustawienie zmiennej odpowiedzialnej za pierwszą grę

```
case SNAKE_KEY_RESET:
                        //rysowanie węża
                        SNAKE_DrawArea();
                        //ustawienie pozycji defaultowej
                        SNAKE_SetDefaultSnake();
                        //generowanie cukierka
                        SNAKE_GenerateTarget();
                        //reset flagi game over z poprzedniej rozgrywki
                        GameOver = 0;
                        //resetowanie flagi czasu
                        Snake_FirstTime = 1;
                        break;
```

Obsługa podczas zjedzenia cukierka:
zwiększenie wyniku struktury węża
printowanie na wyświetlacza wyniku w polu informacyjnym

```
if (Snake1.Hits != Snake.Hits) {
            //zapisanie wyniku
            Snake1.Hits = Snake.Hits;
            //wyświetlenie prędkości
                sprintf(Buffer, "Hits:%4d; Score: %5d", Snake.Hits, (2 -
Settings.Overflow) * Snake.Hits * Settings.Speed);
               ILI9341_Puts(10, SNAKE_TEXT_LINE2, Buffer, &TM_Font_7x10,
0x0000, SNAKE_COLOR_MAIN_BCK);
        }
```

Funkcja odpowiada za rysowanie węża sie ekranie w momencie odświeżenia się gry następuje nadpisanie się pola z tyłu węża. na kolor mapy w przypadku zjedzenia cukierka następuje nadpisanie pola węża na jego głowę. Ostatnie 2 warunki odpowiadają za rysowanie nowej głowy albo czyszczenie starych pikseli. Metoda ILI9341_DrawFilledRectangle jest wywoływana z systemowej biblioteki obsługi wyświetlacza LCD i w tym kiejscu odpowiada za rysowanie segmentów z których wąż jest zbudowany opartych na jego obecnym położeniu.

```
void SNAKE_DrawPixel(uint8_t x, uint8_t y, uint16_t value) {
    uint16_t color;
    //pozyskanie koloru pixela
    if (value == 0) {
        color = SNAKE_COLOR_GAME_BCK;    //wyczyszczenie piksela
    } else if (value == 1) {
        color = SNAKE_COLOR_PIXEL;       //ustawienie piksela
    } else if (value == 2) {
        color = SNAKE_COLOR_TARGET;      //ustawienie docelowego piksela
    } else if (value == 3) {
        color = SNAKE_COLOR_HEAD;        //ustawienie głowy węża
    }

    //rysowanie wypełnionego prostokąta
    ILI9341_DrawFilledRectangle(
        SNAKE_BACK_START_X + 2 + x * (SNAKE_PIXEL_SIZE + 1),
        SNAKE_BACK_START_Y + 2 + y * (SNAKE_PIXEL_SIZE + 1),
            SNAKE_BACK_START_X + 1 + x * (SNAKE_PIXEL_SIZE + 1) +
SNAKE_PIXEL_SIZE,
            SNAKE_BACK_START_Y + 2 + y * (SNAKE_PIXEL_SIZE + 1) +
SNAKE_PIXEL_SIZE,
        color
    );
}
```