KMT AMU 2019
Arduino

Wstęp do programowania
Program podstawowy

```
#include <ISALiquidCrystal.h>
ISALiquidCrystal lcd;

void setup() {
  // put your setup code here, to run once:
  lcd.begin();
  lcd.print("hello, world!");
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Powyższy kod składa się z dwóch funkcji *setup* oraz *loop*. Kod w funkcji *setup*, czyli zawarty między klamrami "{}" zostanie wykonany jeden raz, podczas uruchomienia Arduino. Kod w funkcji *loop* – między drugą parą klamer "{}" będzie wykonywany wielokrotnie aż do wyłączenia Arduino.

Instrukcja warunkowa **if**

Instrukcja **if** wygląda następująco:

```
if (expression)
{
       instruction_1;
       instruction_2;
       ...
}
```

Jeżeli warunek *expression* jest spełniony (prawdziwy), wtedy instrukcje *instruction_1*, *instruction_2*, itd. są wykonywane. Po każdej instrukcji musi być postawiony średnik (**;**). Jeżeli warunek *expression* nie jest spełniony (jest fałszywy), wtedy instrukcje nie zostaną wykonane.

Operator porównanie ==. Zwraca prawdę, jeżeli obiekt po lewej stronie operatora jest równy obiektowy po prawej, a fałsz w przeciwnym przypadku.

Przykład:

```
#include <ISALiquidCrystal.h>
ISALiquidCrystal lcd;

void setup() {
  // put your setup code here, to run once:
  lcd.begin();
  if(1==1){
       lcd.print("1 equal to 1");
  }
```

```
  if(1==3){
       lcd.print("1 equal to 3");
  }
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Funkcje ogólne:

delay(ms) - umożliwia zatrzymanie aktualnie wykonywanego programu na określony czas. *ms* - czas w milisekundach, na jaki ma zostać zatrzymany program.

digitalWrite(pin,value) - umożliwia ustawienie stanu pinu. *pin* – numer pinu, którego stan ma zostać ustawiony, *value* – stan pinu, może przyjąć jedną z dwóch wartości: HIGH lub LOW

- StałaHIGH oznacza diodę lub brzęczyk włączony.
- StałaLOW oznacza diodę wyłączoną lub brzęczyk wyciszony.

Płytka edukacyjna posiada osiem diod LED, oznaczonych symbolami **LED1**, **LED2**, ..., **LED7**, **LED8** oraz brzęczyk **BUZZER**.

digitalRead(pin) - umożliwia ustawienie stanu pinu. *pin* – numer pinu, którego stan ma zostać odczytany. Funkcja będzie wykorzystana do pobrania stanu przycisków oznaczonych symbolami **KEY_UP**, **KEY_DOWN**, **KEY_RIGHT** i **KEY_LEFT**. Jeżeli przycisk jest wciśnięty funkcja zwraca wartość **false** (fałsz), w przeciwnym przypadku **true** (prawda).

Labirynt:

Twoim zadaniem jest znalezienie wyjścia z labiryntu. Aby dotrzeć do wyjścia musisz pokierować ruchem postaci za pomocą odpowiednich komend i uzupełnić poniższy kod:

```
#include "Maze.hpp"

void setup() {
       Maze maze;
       /* Tutaj wstaw swoje rozwiązanie */
}

void loop() {

}
```

Postacią można poruszać za pomocą czterech komend: maze.GoUp(), maze.GoDown(), maze.GoLeft(), maze.GoRight(). Wywołanie danej komendy spowoduje, że postać rozpocznie spacer aż do napotkania przeszkody (ściany) lub wyjścia z labiryntu.
Postać gracza, punkt startu oraz wyjścia oznaczone są na wyświetlanej mapie.

Labirynt 2:

Przygotuj wersję interaktywną gry wyjścia z labiryntu. Gra ma polegać na tym, to gracz (Ty)

sterujesz swoją postacią. Sterowanie może być zrealizowne natępująco: korzystając z funkcji *digitalRead* sprawdź czy klawisz jednego z czterech kierunków jest wciśnięty. Jeśli tak, to wykonaj akcję ruchu w danym kierunku.

Snake:

Twoim zadaniem będzie przygotowanie Arduionowej wersji popularnej gry Snake. We wskazanych fragmentach funkcji loop będziesz musiał/-a dopisać fragmenty kodu, odpowiadające za logikę gry. Podstawowa wersja kodu umożliwia poruszanie się węża w określonym kierunku:

```
#include <ISAOLED.h>
#include <utility>
#include <ISADefinitions.h>
#include <ISA7SegmentDisplay.h>
#include <ISALiquidCrystal.h>
#include "snake.h"

Snake snake;

void setup()
{
    snake.init();
}

void loop()
{
    snake.move_snake();

    snake.display();
    delay(snake.get_speed());
}
```

Wykorzystując przygotowane przez nas funkcje, będziesz miał/-a za zadanie umożliwić:

- zmianę kierunku poruszanie się węża,
- zwiększanie jego długości,
- wyświetlanie zdobytej liczby punktów,
- sprawdzanie, czy wąż najechał na siebie,
- zwiększanie poziomu trudności.

Funkcje ogólne:

snake.change_direction(dir) - metoda zmienia kierunek ruchy węża, przyjmuje jako parametr *dir* nowy kierunek – **UP**, **DOWN**, **LEFT** lub **RIGHT**.
Przykładowe wywołanie:

```
snake.change_direction(UP);
```

snake.get_head_position() - metoda zwraca pozycję głowy węża.

snake.get_food_position() - metoda zwraca pozycję jedzenia.

snake.draw_food() - metoda losuje nowe współrzędne jedzenia.

snake.eat() - metoda informuje węża, że zjadł posiłek.

snake.display_points() - metoda wyświetla aktualną liczbę punktów zdobytych przez gracza na

wyświetlaczu siedmiosegmentowym.

snake.check_uroboros_state() - metoda sprawdza czy przypadkiem wąż nie zjada sam siebie.

snake.restart() - metoda rozpoczyna rozgrywkę od nowa.

snake.change_speed(speed) - metoda zmienia szybkość rozgrywki, przyjmuje wartości od 0 do 8. Przykładowe wywołanie:

```
snake.change_speed(4);
```

snake.restart() - metoda zwraca liczbę punktów zdobytych przez gracza.

Kod do uzupełnienia:

Pod każdym zakomentowanym fragmentem kodu (tekst znajdujący się pomiędzy znakami /* i */) Twoim zadaniem będzie wstawienie odpowiednich fragmentów kodu, wykorzystując funkcje opisane powyżej.

```
#include <ISAOLED.h>
#include <utility>
#include <ISADefinitions.h>
#include <ISA7SegmentDisplay.h>
#include <ISALiquidCrystal.h>
#include "snake.h"

Snake snake;

void setup()
{
    snake.init();
}

void loop()
{

    /*
    Dodaj fragment kodu umożliwiający zmianę kierunku poruszania się węża poprzez
naciśnięcie przycisku.
    Podpowiedź:
        wykorzystaj instrukcję warunkową if oraz metodę snake.change_direction();
    */

    snake.move_snake();

    /*
     Sprawdź, czy głowa snake'a znajduje się w tym samym miejscu co owoc. Jeżeli głowa węża
znajdzie się w tym samym miejscu co owoc, wydłuż węża oraz wylosuj nową pozycję owoca.
     Podpowiedź:
        wykorzystaj instrukcję warunkową if oraz metody:
            snake.get_head_position();
            snake.get_food_position();
            snake.draw_food();
            snake.eat();
     */

    /*
     Wyświetl aktualną liczbę punktów zdobytą przez gracza.
     */

    /*
     Sprawdź, czy wąż nie zaczął przypadkiem zjadać swojego ogona. Jeżeli tak, uruchom
```

```
brzęczyk na 100 milisekund,
     a następnie odczekaj sekundę i rozpocznij rozgrywkę od nowa.
     Podpowiedź:
     wykorzystaj instrukcję warunkową if oraz metody:
         snake.check_uroboros_state();
         snake.restart();
         delay();
         digitalWrite();
    */

    /*
     Po każdych 4 punktach zdobytych przez gracza, zwiększ szybkość rozgrywki.
     Podpowiedź:
     wykorzystaj metody:
         snake.get_points();
         snake.change_speed();
    */

    /*
     Wyświetl aktualny poziom gracza, właczając odpowiednią liczbę diod.
     Podpowiedź:
     wykorzystaj metody:
         snake.get_points();
         digitalWrite();
    */

    snake.display();
    delay(snake.get_speed());
}
```