**Project\_2 : Airplane Gears**

**main.c**

/\*

Name : Nestin Gregorios Sunny

Date : 09.07.2025

Description :

Project 2 - Implement code for controlling the retraction and extension of Airplane's Landing Gears

\*/

#include"gears.h"

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main()

{

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("            Airplane Gears Project\n");

    printf("                                                By,\n");

    printf("                        Nestin Gregorios Sunny\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    Initiate\_State\_Machine();       //function call

    while (1)

    {

        statetable[current\_state]();

    }

    return 0;

}

**gears.h**

#ifndef GEARS\_H\_

#define GEARS\_H\_

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Enum for State\_Type

typedef enum State

{

    GEAR\_DOWN,

    CHECKING\_BEFORE\_TAKEOFF,

    RAISING\_GEAR,

    GEAR\_UP,

    CHECK\_BEFORE\_LANDING,

    LOWERING\_GEAR,

} State\_Type;

// Enum for Switch\_status

typedef enum Switch

{

    on,

    off // plane is on ground

} Switch\_status;

// Enum for pilot\_lever

typedef enum pilot\_lever

{

    Raising,

    falling

} pilot\_lever;

// Enum for hydraulic\_mechanism

typedef enum hydraulic\_mechanism

{

    working,

    not\_working

} hydraulic\_mechanism;

// Function declarations

void GearDown(void);

void CheckingBeforeTakeOFF(void);

void RaisingGear(void);

void GearUp(void);

void CheckBeforeLanding(void);

void LoweringGear(void);

void Initiate\_State\_Machine(void);

// Function pointer array

extern void (\*statetable[])(void);

// Declaring variables as volatile for Switch\_status, pilot\_lever, hydraulic\_mechanism

extern volatile Switch\_status squat\_switch;

extern volatile Switch\_status limit\_switch;

extern volatile pilot\_lever pl;

extern volatile hydraulic\_mechanism hm;

// Declaring a variable to access State\_type

extern State\_Type current\_state;

// Structure for State\_Table

typedef struct State\_Table

{

    char\* current\_state\_indication;

    char\* light;

    char\* direction\_valve\_status;

    char\* landing\_gear\_hydraulic\_control;

    char\* Gas\_Pressurized\_Spring\_system\_status[2];

} State\_Table;

// Initialize State\_Machine array

extern State\_Table State\_Machine[6];

#endif /\* GEARS\_H\_ \*/

**gears.c**

#include "gears.h"

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

State\_Type current\_state;

volatile Switch\_status squat\_switch;

volatile Switch\_status limit\_switch;

volatile pilot\_lever pl;

volatile hydraulic\_mechanism hm;

void (\*statetable[])(void) =

{

    GearDown,

    CheckingBeforeTakeOFF,

    RaisingGear,

    GearUp,

    CheckBeforeLanding,

    LoweringGear

};

State\_Table State\_Machine[6] =

{

    {"GearDown", "Green", "Down", "Enabled", {"Disabled", "Disabled"}},

    {"CheckingBeforeTakeOFF", "Green", "Down", "Enabled", {"Disabled", "Disabled"}},

    {"RaisingGear", "Red", "Up", "Enabled", {"Enabled", "Disabled"}},

    {"GearUp", "Off", "Up", "Disabled", {"Enabled", "Disabled"}},

    {"CheckBeforeLanding", "Red", "Down", "Enabled", {"Disabled", "Enabled"}},

    {"LoweringGear", "Green", "Down", "Enabled", {"Disabled", "Enabled"}}

};

//function definitions

void GearDown(void)

{

    current\_state = GEAR\_DOWN;      //enum = GEAR\_DOWN

    printf("\nEnter the status of pilot's lever :-\n(0 for Raising, 1 for falling) : ");

    int input;

    if (scanf("%d", &input) != 1)       // 0

    {

        printf("Invalid input\n");

        return;

    }

    pl = (pilot\_lever)input;

    if(pl == Raising)

    {

        current\_state = CHECKING\_BEFORE\_TAKEOFF;        //enum = CHECKING\_BEFORE\_TAKEOFF

        printf("\nCurrent State : %s", State\_Machine[current\_state].current\_state\_indication);

        printf("\nLight is : %s", State\_Machine[current\_state].light);

        printf("\nLanding Gear Hydraulic Control : %s\n", State\_Machine[current\_state].landing\_gear\_hydraulic\_control);

    }

    else

    {

        GearDown();

    }

}

void CheckingBeforeTakeOFF(void)

{

    current\_state = CHECKING\_BEFORE\_TAKEOFF;        //enum = CHECKING\_BEFORE\_TAKEOFF

    printf("\nEnter the status of pilot's lever :-\n(0 for Raising, 1 for falling), squat switch (0 for on, 1 for off) and hydraulic mechanism (0 for working, 1 for not working) : ");

    int input\_pl, input\_squat\_switch, input\_hm;

    if (scanf("%d %d %d", &input\_pl, &input\_squat\_switch, &input\_hm) != 3)      //0 0 0

    {

        printf("Invalid input\n");

        return;

    }

    pl = (pilot\_lever)input\_pl;

    squat\_switch = (Switch\_status)input\_squat\_switch;

    hm = (hydraulic\_mechanism)input\_hm;

    if(pl == falling && squat\_switch == off)

    {

        current\_state = GEAR\_DOWN;      //enum = GEAR\_DOWN

        GearDown();

    }

    else if(pl == Raising && squat\_switch == on && hm == working)

    {

        current\_state = RAISING\_GEAR;       //enum = RAISING\_GEAR

        printf("\nCurrent State : %s", State\_Machine[current\_state].current\_state\_indication);

        printf("\nLight is : %s", State\_Machine[current\_state].light);

        printf("\nStatus of Direction valve : %s\n", State\_Machine[current\_state].direction\_valve\_status);

    }

    else if(pl == Raising && squat\_switch == on && hm == not\_working)

    {

        current\_state = RAISING\_GEAR;       //enum = RAISING\_GEAR

        printf("\nCurrent State : %s", State\_Machine[current\_state].current\_state\_indication);

        printf("\nLight is : %s", State\_Machine[current\_state].light);

        printf("\nStatus of Direction valve : %s", State\_Machine[current\_state].direction\_valve\_status);

        printf("\nStatus of gas pressurised Spring System : %s\n", State\_Machine[current\_state].Gas\_Pressurized\_Spring\_system\_status[0]);

    }

}

void RaisingGear(void)

{

    current\_state = RAISING\_GEAR;       //enum = RAISING\_GEAR

    printf("\nEnter the status of pilot's lever :-\n(0 for Raising, 1 for falling) and limit switch (0 for on, 1 for off) : ");

    int input\_pl, input\_limit\_switch;

    if (scanf("%d %d", &input\_pl, &input\_limit\_switch) != 2)        //0 1

    {

        printf("Invalid input\n");

        return;

    }

    pl = (pilot\_lever)input\_pl;

    limit\_switch = (Switch\_status)input\_limit\_switch;

    if(pl == falling && limit\_switch == on)

    {

        current\_state = CHECK\_BEFORE\_LANDING;   //enum = CHECKING\_BEFORE\_LANDING

        printf("\nCurrent State : %s", State\_Machine[current\_state].current\_state\_indication);

        printf("\nLight is : %s", State\_Machine[current\_state].light);

        printf("\nLanding Gear Hydraulic Control : %s\n", State\_Machine[current\_state].landing\_gear\_hydraulic\_control);

    }

    else if(pl == Raising && limit\_switch == off)

    {

        current\_state = GEAR\_UP;        //enum = GEAR\_UP

        printf("\nCurrent State : %s", State\_Machine[current\_state].current\_state\_indication);

        printf("\nLight is : %s", State\_Machine[current\_state].light);

        printf("\nLanding Gear Hydraulic Control : %s\n", State\_Machine[current\_state].landing\_gear\_hydraulic\_control);

    }

    else

    {

        RaisingGear();

    }

}

void GearUp(void)

{

    current\_state = GEAR\_UP;        //enum = GEAR\_UP

    printf("\nEnter the status of pilot's lever :-\n(0 for Raising, 1 for falling) : ");

    int input;

    if (scanf("%d", &input) != 1)       //1

    {

        printf("Invalid input\n");

        return;

    }

    pl = (pilot\_lever)input;

    if(pl == falling)

    {

        current\_state = CHECK\_BEFORE\_LANDING;       //enum = CHECK\_BEFORE\_LANDING

        printf("\nCurrent State : %s", State\_Machine[current\_state].current\_state\_indication);

        printf("\nLight is : %s", State\_Machine[current\_state].light);

        printf("\nLanding Gear Hydraulic Control : %s\n", State\_Machine[current\_state].landing\_gear\_hydraulic\_control);

    }

    else

    {

        GearUp();

    }

}

void CheckBeforeLanding(void)

{

    current\_state = CHECK\_BEFORE\_LANDING;       //enum = CHECKING\_BEFORE\_LANDING

    printf("\nEnter the status of pilot's lever :-\n(0 for Raising, 1 for falling) and hydraulic mechanism (0 for working, 1 for not working) : ");

    int input\_pl, input\_hm;

    if (scanf("%d %d", &input\_pl, &input\_hm) != 2)      //1 0

    {

        printf("Invalid input\n");

        return;

    }

    pl = (pilot\_lever)input\_pl;

    hm = (hydraulic\_mechanism)input\_hm;

    if(pl == falling && hm == working)

    {

        current\_state = LOWERING\_GEAR;      //enum = LOWERING\_GEAR

        printf("\nCurrent State : %s", State\_Machine[current\_state].current\_state\_indication);

        printf("\nLight is : %s", State\_Machine[current\_state].light);

        printf("\nStatus of Direction valve : %s\n", State\_Machine[current\_state].direction\_valve\_status);

    }

    else if(pl == falling && hm == not\_working)

    {

        current\_state = LOWERING\_GEAR;      //enum = LOWERING\_GEAR

        printf("\nCurrent State : %s", State\_Machine[current\_state].current\_state\_indication);

        printf("\nLight is : %s", State\_Machine[current\_state].light);

        printf("\nStatus of Direction valve : %s", State\_Machine[current\_state].direction\_valve\_status);

        printf("\nStatus of gas pressurised Spring System : %s\n", State\_Machine[current\_state].Gas\_Pressurized\_Spring\_system\_status[1]);

    }

    else if(pl == Raising)

    {

        CheckBeforeLanding();

    }

}

void LoweringGear(void)

{

    current\_state = LOWERING\_GEAR;      //enum = LOWERING\_GEAR

    printf("Enter the status of pilot's lever :-\n(0 for Raising, 1 for falling) and limit switch (0 for on, 1 for off): ");

    int input\_pl, input\_limit\_switch;

    if (scanf("%d %d", &input\_pl, &input\_limit\_switch) != 2)        //1 0

    {

        printf("Invalid input\n");

        return;

    }

    pl = (pilot\_lever)input\_pl;

    limit\_switch = (Switch\_status)input\_limit\_switch;

    if(pl == falling && limit\_switch == on)

    {

        current\_state = CHECKING\_BEFORE\_TAKEOFF;        //enum = CHECKING\_BEFORE\_TAKEOFF

        printf("Current State : %s\n", State\_Machine[current\_state].current\_state\_indication);

        printf("Light is : %s\n", State\_Machine[current\_state].light);

    }

    else if(pl == Raising && limit\_switch == off)

    {

        current\_state = GEAR\_DOWN;      //enum = GEAR\_DOWN

        printf("Current State : %s\n", State\_Machine[current\_state].current\_state\_indication);

        printf("Light is : %s\n", State\_Machine[current\_state].light);

        printf("Landing Gear Hydraulic Control : %s\n", State\_Machine[current\_state].landing\_gear\_hydraulic\_control);

    }

    else if(limit\_switch == off)

    {

        LoweringGear();

    }

}

void Initiate\_State\_Machine()

{

    current\_state = GEAR\_DOWN;      //enum = GEAR\_DOWN

    printf("The Finite State Machine is Intialized and current the program is in GearDown State & Light is : %s\n",State\_Machine[current\_state].light);

}