Dílenská praxe

A4	6. Výtah		
Petřík Vít		1/8	Známka:
22. 1. 2020	Datum odevzdání:	19. 2. 2020	Odevzdáno:

Zadání:

Zpracujte program v programovacím jazyce C ovládající model výtahu tak, aby obsahoval nejméně tyto funkce:

1) ovládání pohybu kabiny výtahu pomocí tlačítek na patrech
2) ovládání pohybu kabiny výtahu pomocí tlačítek v kabině
3) ovládání pomocných funkcí výtahu
4) respektování funkcí tlačítek výtahu v závislosti na stavu výtahu (obsazená, případně plná kabina, . . .)
5) sledování provozních a chybových stavů

Postup:

- Návrh zapojení
- Nadefinování datové struktury.
- Zapsání pater a odpovídajících tlačítek a senzorů do datové struktury.
- Zápis hlavní řídící logiky.
- Návrh a implementace funkce paměti.

Propojení PC a výtahu:

PORT 1 - output		PORT 2 - output		PORT 3 – input*		PORT 4 - input		
Pin	Význam	Pin	Význam	Pin	Význam	Pin	Význam	
0	A	0	MZ	0	1	0	1	Sensory pater
1	B	1	MS	1	1	1	2	
2	C	2	SK	2	2	2	3	
3	-	3	ZZ	3	2	3	4	
4	-	4	UP	4	3	4	DP	
5	-	5	DOWN	5	3	5	PS	
6	-	6	-	6	4	6	IS	
7	-	7	-	7	4	7	-	

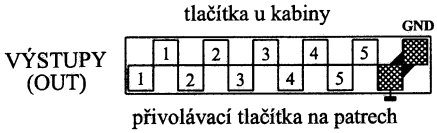
**lichý bity - tlačítka na patře, sudý bity – tlačítka u kabiny*

Schéma zapojení:

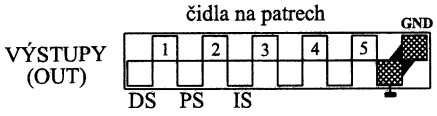
MODEL OSOBNÍHO VÝTAHU

Napájení: 12Vss/1A
Vstupy a Výstupy: všechny jsou v úrovních TTL, aktivní v log. 0
Poznámka: 1) přivolávací tlačítka a čidla pater jsou bezkontaktní - ošetřené proti zákmitům, ostatní spínače je nutno ošetřit softwarově
2) koncové polohy vozíku s kabinou elektronika výtahu nehlídá - nutno hlídat softwarově
3) kabině nikdy "nepomáhejte" rukou!

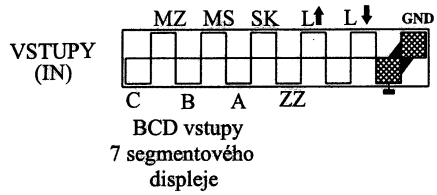
Zapojení vývodů:



AKTIVNÍ = LOG. 0



DS = dveřní spínač kabiny
PS = podlahový spínač kabiny
IS = impulsní snímač otáčení na hřídeli pohybu kabiny



MZ = zapnutí otáčení motoru
MS = směr otáčení motoru 0 =↓, 1= ↑
SK = zapnutí světla kabiny
L↑ L↓ = LED u přivolávacích tlačítek
C, B, A = zobrazení čísla patra (v BCD kódu)
ZZ = zvukové upozornění

Vývojový diagram:

Příloha 1

Výpis programu:

Příloha 2 – C++

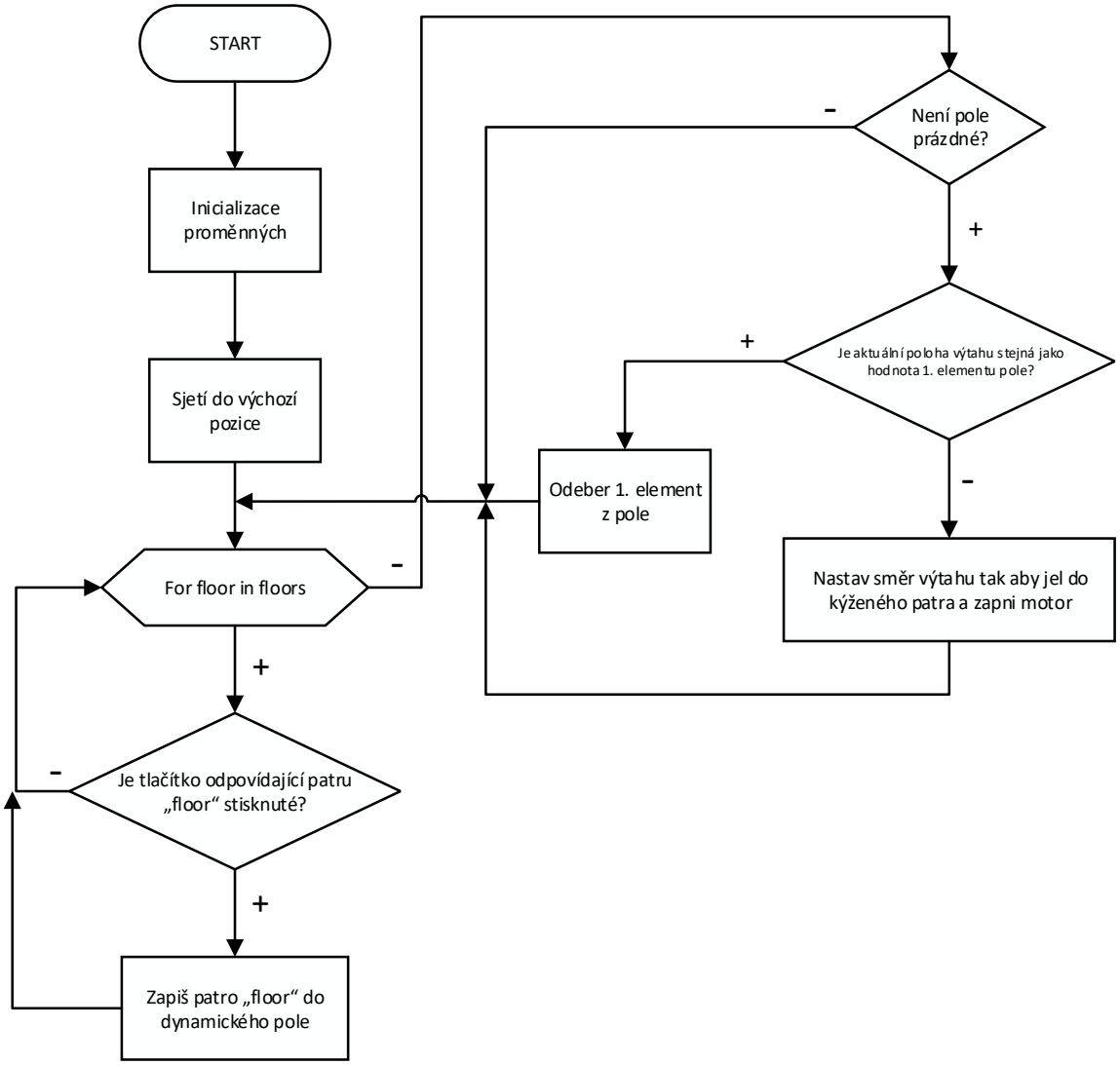
Závěr:

Program funguje tak jak má a je spolehlivý. V programu využívám konstrukci s dynamickým polem pro ukládání pater na která má výtah dojet.

Přílohy:

- Příloha 1 – 1 strana
- Příloha 1 – 4 strany

Příloha 1


START
Inicializace proměnných
Sjetí do výchozí pozice
For floor in floors
Je tlačítko odpovídající patru „floor“ stisknuté?
Zapiš patro „floor“ do dynamického pole
Není pole prázdné?
Je aktuální poloha výtahu stejná jako hodnota 1. elementu pole?
Odeber 1. element z pole
Nastav směr výtahu tak aby jel do kýženého patra a zapni motor
+
-

Příloha 2

```
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <stdlib.h>

#define P1 0x300
#define P2 0x301
#define P3 0x300
#define P4 0x301

//vrati pocet milisekund od startu programu
unsigned long millis()
{
    return (clock() * 1000) / CLOCKS_PER_SEC;
}

typedef struct
{
    char port;
    char bit;
} Pin;

typedef struct
{
    Pin cabinBtn;
    Pin floorBtn;
    Pin sensor;
    char keyBoard;
} Floor;

typedef struct
{
    Pin conn;
    char state;
} Sensor;

typedef struct
{
    Sensor door;
    Sensor floor;
    Sensor revolutions;
} Sensors;

//pointer do ktereho hazime patra do kterych ma vytah dojet
char *ptr = (char *)malloc(sizeof(char) * 0);
long size = 0;

const Floor floors[] = {
    {{2, 0}, {2, 1}, {3, 0}, '1'},
    {{2, 2}, {2, 3}, {3, 1}, '2'},
    {{2, 4}, {2, 5}, {3, 2}, '3'},
    {{2, 6}, {2, 7}, {3, 3}, '4'},
};

Sensors sensors = {{{3, 4}, 0}, {{3, 5}, 0}, {{3, 6}, 0}};
```

```
//prida element do dynamickeho pole
long addFloor(char floor)
{
    //pokud je posledni podlazi v poli stejne jako to, ktere chceme zapsat
    //vykasleme se na to a opustime funkci
    if (*(ptr + size - 1) == floor && size != 0)
    {
        return;
    }
    size++;

    //realokujeme blok pameti s jinou velikosti
    char *newPointer = (char *)realloc(ptr, sizeof(char) * size);
    if (newPointer != NULL)
    {
        //pokud je vse v poradku
        ptr = newPointer;
        *(ptr + size - 1) = floor;
    }

    //vypiseme patra na ktera vytah pojede
    printf("\n\rVytah pojede na nasledujicich pater (velikost pole: %i):", size);
    for (long j = 0; j < size; j++)
    {
        printf(" %i", *(ptr + j) + 1);
    }
}

//odebere 1. element z dynamickeho pole
void removeFirst()
{
    //pokud ma pole nejakou velikost
    if (size)
    {
        //posuneme vsechna data o jedno doleva
        for (int i = 1; i < size; i++)
        {
            *(ptr + i - 1) = *(ptr + i);
        }
        size--;

        //realokujeme pamet
        char *newPointer = (char *)realloc(ptr, sizeof(char) * size);
        if (newPointer != NULL)
        {
            ptr = newPointer;
        }

        //vypiseme patra na ktera vytah pojede
        printf("\n\rVytah pojede na nasledujicich pater (velikost pole: %i):", size);
        for (long j = 0; j < size; j++)
        {
            printf(" %i", *(ptr + j) + 1);
        }
    }
}
```

```
int main(void)
{
    //inicializace promennych
    char cabinFloor = 0;
    char pressedKey;
    unsigned long lightMillis = 0;
    unsigned long lastMoveMillis = 0;
    addFloor(3);
    int ports[4];
    while (1)
    {
        //vezme stisknutou klavesu (pokud nejake je)
        if (kbhit())
        {
            pressedKey = getch();
        }
        //pokud ne tak vynulujeme promennou
        else
        {

            pressedKey = NULL;
        }

        //precteni portu a prevedeni vystupnich portu do neaktivni hodnoty
        ports[0] = 0xFF;
        ports[1] = 0xFF;
        ports[2] = inportb(P3);
        ports[3] = inportb(P4);

        //stav sensoru
       sensors.door.state = !(ports[sensors.door.conn.port] & (1 << sensors.door.conn.bit)) ? 1 : 0;
       sensors.floor.state = (ports[sensors.floor.conn.port] & (1 << sensors.floor.conn.bit)) ? 1 : 0;

        //pokud jsou dvere otevrena nebo je aktivovan podlahovy sensor zapneme svetylko
        lightMillis = sensors.floor.state || !sensors.door.state ? millis() : lightMillis;

        //proskenujeme vsechna podlazi
        for (char i = 0; i < sizeof(floors) / sizeof(Floor); i++)
        {
            //zjisti jestli je dane patro aktivni
           cabinFloor = !(ports[floors[i].sensor.port] & 1 << floors[i].sensor.bit) ? i : cabinFloor;

            //precte stav tlacitka u kabiny
           if (!(ports[floors[i].cabinBtn.port] & 1 << floors[i].cabinBtn.bit) && sensors.floor.state)
            {
                lightMillis = millis();
                addFloor(i);
            }

            //zjisti stav privolavacich tlacitek
            if (!(ports[floors[i].floorBtn.port] & (1 << floors[i].floorBtn.bit)) ||
                                                        floors[i].keyBoard == pressedKey)
            {
                lightMillis = millis();
                addFloor(i);
            }
        }
```

```
        //pokud jsou v poli nejake hodnoty znamena to, ze se s vytahem ma neco stat
        if (size)
        {
            //pokud je patro kabiny stejne jako patro do ktereho mame jet
            if (*(ptr) == cabinFloor)
            {
                //na dalsi pohyb pockame 2s aby clovek stihl vystoupit
                if (millis() - lastMoveMillis > 2000)
                {
                    removeFirst();
                }
                //Na chvilku zapneme pipak
                if (millis() - lastMoveMillis < 100)
                {
                    ports[1] &= ~(1 << 3); //zapnuti pipaku
                }
            }
            //pokud jsou dvere zavrena
            else if (sensors.door.state)
            {
                ports[1] &= ~(1 << 0);                              //zapnuti motoru
                ports[1] &= ~((*(ptr) < cabinFloor ? 1 : 0) << 1); //nastaveni smeru
                ports[1] &= ~(1 << (*(ptr) < cabinFloor ? 5 : 4)); //zapnuti signalizace smeru
                lightMillis = millis();
                lastMoveMillis = millis();
            }
        }
        ports[1] &= ~(((millis() - lightMillis) < 2000 ? 1 : 0) << 2); //zapnuti svetylka
        ports[0] = (cabinFloor + 1) | 0xF8;                             //nastaveni BCD kodu

        //Jachyme hod to do stroje
        outportb(P1, ports[0]);
        outportb(P2, ports[1]);

        //pokud zmackneme ESC hodime do portu neaktivni hodnotu a opustime smycku
        if (pressedKey == 27)
        {
            outportb(P1, 0xFF);
            outportb(P2, 0xFF);
            break;
        }
        //pokud zmacknem "DELETE" vymazeme prvni element pole
        else if (pressedKey == 83)
        {
            removeFirst();
        }
    }
    //konec programu a navrat do Borlanda
    while (!kbhit())
        ;
    return 0;
}
```