Dílenská praxe

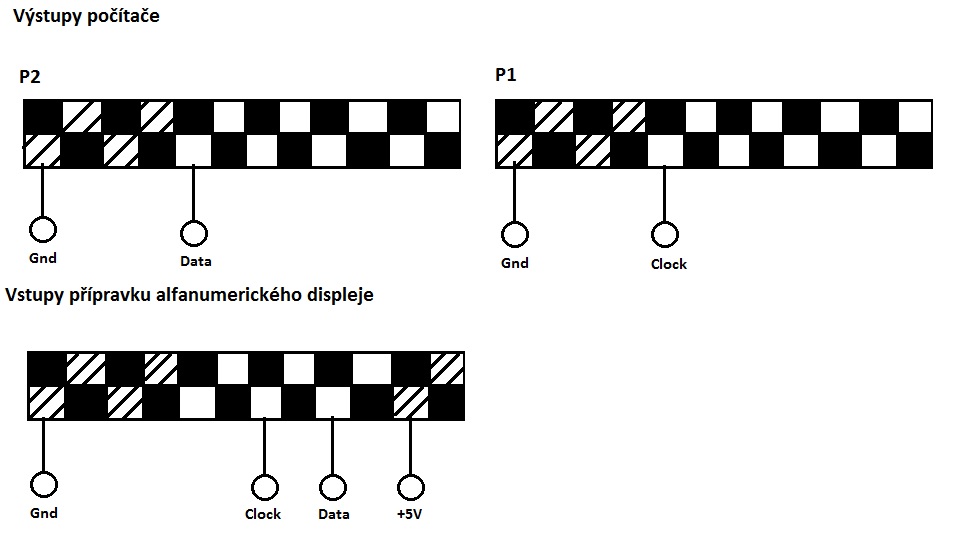
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A4** | C. Alfanumerický displej | | | |
| Macháček Daniel | |  | 1/22 | Známka: |
| 26.10.2016 | | Datum odevzdání: | 16.11.2016 | Odevzdáno: |

Zadání:

Zpracujte program v programovacím jazyce C ovládající alfanumerický displej tak, aby obsahoval nejméně tyto funkce:

1. volbu druhu displeje (7segmentový/14segmentový)
2. zobrazení vhodně zvolené množiny znaků pro každý typ displeje
3. vhodně zvolená datová a programová struktura

**Zapojení vstupů a výstupů:**



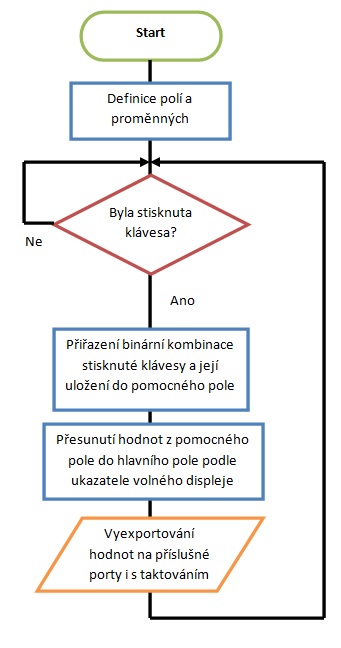
Závěr:

Při testování tohoto programu, jsem měl opakovaně chyby v zápisu do pole. Kvůli následnému spěchu v zájmu úlohu včas odevzdat, jsem využil prvního nápadu, který vedl k odstranění chyby. Určitě existují mnohem lepší a efektivnější nápady, ale neměl jsem čas se jimi zaobírat. Program umožňuje rozsvítit na displeji číslice 0-9 (v případě numerického displeje) a k tomu ještě určitá písmena abecedy (v případě alfanumerického displeje). Jedná se o písmena A-Z s vynecháním diakritiky. Na alfanumerickém displeji jsou číslice zobrazovány s tečkou z důvodu odlišení od některých, podobně vypadajících písmen. Znaky se na displejích zobrazují zleva doprava (první znak na první displej, druhý znak na druhý displej…) a při naplnění všech se celý displej anuluje a začíná od začátku (zleva). Vytvořil jsem program pro každý displej zvlášť.

Přílohy:

* Výpis programu – 4 strany
* Vývojový diagram – 19 stran

**Vývojový Diagram:**



**Výpis programu numerického displeje:**

#include <stdio.h> //Otevření knihovny

#include <dos.h> //Otevření knihovny

int pointer = 1; //Definice proměnné

int status = 1;

int a,b,c,d,e;

int stisk;

int pole[36]; //Definice pole o velikosti 36 bitů

int binary[7];

int binar0[7] = {1 , 1 , 1 , 1 , 1 , 1 , 0}; //Definice pole o velikosti 7 bitů a načtení jednotlivých bitů

int binar1[7] = {0 , 1 , 1 , 0 , 0 , 0 , 0};

int binar2[7] = {1 , 1 , 0 , 1 , 1 , 0 , 1};

int binar3[7] = {1 , 1 , 1 , 1 , 0 , 0 , 1};

int binar4[7] = {0 , 1 , 1 , 0 , 0 , 1 , 1};

int binar5[7] = {1 , 0 , 1 , 1 , 0 , 1 , 1};

int binar6[7] = {1 , 0 , 1 , 1 , 1 , 1 , 1};

int binar7[7] = {1 , 1 , 1 , 0 , 0 , 0 , 0};

int binar8[7] = {1 , 1 , 1 , 1 , 1 , 1 , 1};

int binar9[7] = {1 , 1 , 1 , 1 , 0 , 1 , 1};

void main() //Hlavní pole

{

while(1)

{

switch(status) //Nekonečný cyklus

{

case 1: //Volba čísla

{

printf("Dostupna cisla: 0,1,2,3,4,5,6,7,8,9\n"); //Vypíše větu uvedenou v závorce

printf("Stiskni cislo: "); //Vypíše větu uvedenou v závorce

scanf("%i", &stisk); //Testuje zda je nějaká klávesa stisknuta

if(stisk >= 0 && stisk < 10) //Pokud byla stisknuta klávesa v uvedeném intervalu

{

pole[0] = 1; //Vypíše na nultou pozici v poli 1

status = 2; //Nastaví proměnnou status hodnotou 2

}

else

{

printf("/n Nevybrali jste ze zadanych cisel. /n"); //Vypíše větu uvedenou v závorce

status = 1; //Nastaví proměnnou status hodnotou 1

}

}

break; //Ukončení ůkonu

case 2: //Přiřazení binární hodnoty dle stisknuté klávesy

{

switch(stisk) //Výběr procesu dle hodnoty v proměnné stisk

{

case 0: //Byla stisknuta 0

{

for(d = 0; d <= 6; d++) //Provádí cyklus za dodržení podmínky

{

for(e = 0;e <= 6; e++) //Provádí cyklus za dodržení podmínky

{

binary[d] = binar0[e]; //Přeposílá hodnoty z pole binar0 do pole binary

}

}

}

break; //Ukončení úkonu

case 1:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar1[e];

}

}

}

break;

case 2:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar2[e];

}

}

}

break;

case 3:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar3[e];

}

}

}

break;

case 4:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar4[e];

}

}

}

break;

case 5:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar5[e];

}

}

}

break;

case 6:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar6[e];

}

}

}

break;

case 7:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar7[e];

}

}

}

break;

case 8:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar8[e];

}

}

}

break;

case 9:

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binary[d] = binar9[e];

}

}

}

break;

}

status = 3; //Nastaví proměnnou status hodnotou 3

}

break;

case 3: //Přepnutí na další displej

{

if(pointer == 1) //Pokud se proměnná pointer rovná 1

{

for(a = 1; a <= 35; a++) //Provádí cyklus za dodržení podmínky

{

pole[a] = 0; //Naplnění pole nulami

}

for(a = 1; a <= 7; a++) //Provádí cyklus za dodržení podmínky

{

for(b = 0; b <= 6; b++) //Provádí cyklus za dodržení podmínky

{

pole[a] = binary[b]; //Přeposílá hodnoty z pole binary do hlavního pole

}

}

pointer = pointer + 1; //Přičtení 1 do proměnné pointer

}

if(pointer == 2)

{

for(a = 8; a <= 14; a++)

{

for(b = 0; b <= 6; b++)

{

pole[a] = binary[b];

}

}

pointer = pointer + 1;

}

if(pointer == 3)

{

for(a = 15; a <= 21; a++)

{

for(b = 0; b <= 6; b++)

{

pole[a] = binary[b];

}

}

pointer = pointer + 1;

}

if(pointer == 4)

{

for(a = 22; a <= 28; a++)

{

for(b = 0; b <= 6; b++)

{

pole[a] = binary[b];

}

}

pointer = pointer + 1;

}

if(pointer == 5)

{

for(a = 29; a <= 35; a++)

{

for(b = 0; b <= 6; b++)

{

pole[a] = binary[b];

}

}

pointer = 1;

}

status = 4; //Nastaví proměnnou status hodnotou 4

}

break; //Ukončení úkonu

case 4: //Poslání hodnot na port

{

for (c = 1; c <= 35; c++) //Provádí cyklus za dodržení podmínky

{

outportb(0x301, pole[c]); //Poslání hodnoty z hlavního pole na port P2

outportb(0x300, 0x01); //Poslání taktovací hodnoty na port P1

outportb(0x300, 0x00); //Poslání taktovací hodnoty na port P1

}

status = 1; //Nastaví proměnnou status hodnotou 1

}

break; //Ukončení úkonu

}

}

}

**Výpis programu alfanumerického displeje:**

#include <stdio.h> //Otevření knihovny

#include <dos.h> //Otevření knihovny

int pointer = 1; //Definice proměnné

int status = 1;

int a,b,c,d,e;

char stisk;

int binar[15]; //Definice pole o velikosti 15 bitů

int pole1[36]; //Definice pole o velikosti 36 bitů

int pole2[36];

int pole3[36];

int pole4[36];

int binar0[15] = {1 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 1}; //Definice pole o velikosti 15 bitů a načtení jednotlivých bitů

int binar1[15] = {0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1};

int binar2[15] = {1 , 1 , 0 , 1 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1};

int binar3[15] = {1 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1};

int binar4[15] = {0 , 1 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1};

int binar5[15] = {1 , 0 , 1 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1};

int binar6[15] = {1 , 0 , 1 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1};

int binar7[15] = {1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1};

int binar8[15] = {1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1};

int binar9[15] = {1 , 1 , 1 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1};

int binara[15] = {1 , 1 , 1 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0};

int binarb[15] = {1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0};

int binarc[15] = {1 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0};

int binard[15] = {1 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0};

int binare[15] = {1 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0};

int binarf[15] = {1 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0};

int binarg[15] = {1 , 0 , 1 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0};

int binarh[15] = {0 , 1 , 1 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0};

int binari[15] = {1 , 0 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0};

int binarj[15] = {0 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0};

int binark[15] = {0 , 0 , 0 , 0 , 1 , 1 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0};

int binarl[15] = {0 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0};

int binarm[15] = {0 , 1 , 1 , 0 , 1 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0};

int binarn[15] = {0 , 1 , 1 , 0 , 1 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0};

int binaro[15] = {1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0};

int binarp[15] = {1 , 1 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0};

int binarq[15] = {1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0};

int binarr[15] = {1 , 1 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 0};

int binars[15] = {1 , 0 , 1 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0};

int binart[15] = {1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0};

int binaru[15] = {0 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0};

int binarv[15] = {0 , 0 , 0 , 0 , 1 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0};

int binarw[15] = {0 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0};

int binarx[15] = {0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 1 , 0 , 1 , 0};

int binary[15] = {0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 1 , 0};

int binarz[15] = {1 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0};

void main() //Hlavní pole

{

while(1) //Nekonečný cyklus

{

switch(status) //Výběr úkonu dle hodnoty status

{

case 1: //Volba čísla

{

printf("Jsou dostupne vsechny znaky\n"); //Vypíše větu uvedenou v závorce

printf("Vyberte znak: "); //Vypíše větu uvedenou v závorce

if (kbhit()) //"Zachytává" stisknutou klávesu

{

stisk = getch(); //Stisknutou klávesu uloží do proměnné stisk

pole1[0] = 1; //Nultému bitu v pole1 přiřadí hodnotu 1

pole2[0] = 1; //Nultému bitu v pole2 přiřadí hodnotu 1

pole3[0] = 1; //Nultému bitu v pole3 přiřadí hodnotu 1

pole4[0] = 1; //Nultému bitu v pole4 přiřadí hodnotu 1

pole1[31] = 0; //Třicátému prvnímu bitu v pole1 přiřadí hodnotu 0

pole2[31] = 1; //Třicátému prvnímu bitu v pole2 přiřadí hodnotu 1

pole3[31] = 0; //Třicátému prvnímu bitu v pole3 přiřadí hodnotu 0

pole4[31] = 1; //Třicátému prvnímu bitu v pole4 přiřadí hodnotu 1

pole1[32] = 1; //Třicátému druhému bitu v pole1 přiřadí hodnotu 1

pole2[32] = 0; //Třicátému druhému bitu v pole2 přiřadí hodnotu 0

pole3[32] = 1; //Třicátému druhému bitu v pole3 přiřadí hodnotu 1

pole4[32] = 0; //Třicátému druhému bitu v pole4 přiřadí hodnotu 0

status = 2; //Nastaví proměnnou status hodnotou 2

}

else //Pokud nebyla klávesa stisknuta

{

status = 1; //Nastaví proměnnou status hodnotou 1

}

}

break; //Konec procesu

case 2: //Přiřazení binární hodnoty dle stisknuté klávesy

{

switch(stisk) //Výběr procesu dle hodnoty v proměnné stisk

{

case '0': //V případě stisknutí 0

{

for(d = 0; d <= 6; d++) //Provádí cyklus za dodržení podmínky

{

for(e = 0;e <= 6; e++) //Provádí cyklus za dodržení podmínky

{

binar[d] = binar0[e]; //Přeposílá hodnoty z pole binar0 do pole binar

}

}

}

break; //Ukončení úkonu

case '1': //V případě stisknutí 1

{ // - to samé co v předchozím úkonu jen z jiného pole

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar1[e];

}

}

}

break;

case '2':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar2[e];

}

}

}

break;

case '3':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar3[e];

}

}

}

break;

case '4':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar4[e];

}

}

}

break;

case '5':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar5[e];

}

}

}

break;

case '6':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar6[e];

}

}

}

break;

case '7':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar7[e];

}

}

}

break;

case '8':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar8[e];

}

}

}

break;

case '9':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binar9[e];

}

}

}

break;

case 'a':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binara[e];

}

}

}

break;

case 'b':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarb[e];

}

}

}

break;

case 'c':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarc[e];

}

}

}

break;

case 'd':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binard[e];

}

}

}

break;

case 'e':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binare[e];

}

}

}

break;

case 'f':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarf[e];

}

}

}

break;

case 'g':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarg[e];

}

}

}

break;

case 'h':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarh[e];

}

}

}

break;

case 'i':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binari[e];

}

}

}

break;

case 'j':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarj[e];

}

}

}

break;

case 'k':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binark[e];

}

}

}

break;

case 'l':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarl[e];

}

}

}

break;

case 'm':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarm[e];

}

}

}

break;

case 'n':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarn[e];

}

}

}

break;

case 'o':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binaro[e];

}

}

}

break;

case 'p':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarp[e];

}

}

}

break;

case 'q':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarq[e];

}

}

}

break;

case 'r':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarr[e];

}

}

}

break;

case 's':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binars[e];

}

}

}

break;

case 't':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binart[e];

}

}

}

break;

case 'u':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binaru[e];

}

}

}

break;

case 'v':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarv[e];

}

}

}

break;

case 'w':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarw[e];

}

}

}

break;

case 'x':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarx[e];

}

}

}

break;

case 'y':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binary[e];

}

}

}

break;

case 'z':

{

for(d = 0; d <= 6; d++)

{

for(e = 0;e <= 6; e++)

{

binar[d] = binarz[e];

}

}

}

break;

}

status = 3; //Nastaví proměnnou status hodnotou 3

}

break;

case 3: //Přepnutí na další displej

{

if(pointer == 1) //Pokud se proměnná pointer rovná 1

{

for(a = 1; a <= 15; a++) //Provádí cyklus za dodržení podmínky

{

pole1[a] = 0; //Naplnění pole1 nulami

pole2[a] = 0; //Naplnění pole2 nulami

pole3[a] = 0; //Naplnění pole3 nulami

pole4[a] = 0; //Naplnění pole4 nulami

}

for(a = 1; a <= 15; a++) //Provádí cyklus za dodržení podmínky

{

for(b = 0; b <= 14; b++) //Provádí cyklus za dodržení podmínky

{

pole1[a] = binar[b]; //Přeposílá hodnoty z pole binar do hlavního pole1

}

}

pointer = pointer + 1; //Přičtení 1 do proměnné pointer

}

if(pointer == 2)

{

for(a = 1; a <= 15; a++)

{

for(b = 0; b <= 14; b++)

{

pole2[a] = binar[b];

}

}

pointer = pointer + 1;

}

if(pointer == 3)

{

for(a = 16; a <= 30; a++)

{

for(b = 0; b <= 14; b++)

{

pole3[a] = binar[b];

}

}

pointer = pointer + 1;

}

if(pointer == 4)

{

for(a = 16; a <= 30; a++)

{

for(b = 0; b <= 14; b++)

{

pole4[a] = binar[b];

}

}

pointer = 1;

}

status = 4; //Nastaví proměnnou status hodnotou 4

}

break; //Ukončení úkonu

case 4: //Poslání hodnot na port

{

for (c = 0; c <= 35; c++) //Provádí cyklus za dodržení podmínky

{

outportb(0x301, pole1[c]); //Poslání hodnoty z hlavního pole na port P2

outportb(0x300, 0x01); //Poslání taktovací hodnoty na port P1

outportb(0x300, 0x00); //Poslání taktovací hodnoty na port P1

outportb(0x301, pole2[c]); //Poslání hodnoty z hlavního pole na port P2

outportb(0x300, 0x01); //Poslání taktovací hodnoty na port P1

outportb(0x300, 0x00); //Poslání taktovací hodnoty na port P1

outportb(0x301, pole3[c]); //Poslání hodnoty z hlavního pole na port P2

outportb(0x300, 0x01); //Poslání taktovací hodnoty na port P1

outportb(0x300, 0x00); //Poslání taktovací hodnoty na port P1

outportb(0x301, pole4[c]); //Poslání hodnoty z hlavního pole na port P2

outportb(0x300, 0x01); //Poslání taktovací hodnoty na port P1

outportb(0x300, 0x00); //Poslání taktovací hodnoty na port P1

}

status = 1; //Nastaví proměnnou status hodnotou 1

}

break; //Konec úkonu

}

}

}