6882

PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81
ZESPÓŁ AUTOMATYKI ELEKTRONICZNEJ
PRACOWNIA REGULATORÓW ELEKTRONICZNYCH

440 BE 1

Główny wykonawca mgr inż. Zbigniew Pietrusiński

Wykonawcy mgr inż. Grzegorz Kazimierski,
mgr inż. Andrzej Cichy

Konsultant

Nr zlecenia S1307

Badania symulacyjne algorytmów autostrojenia regulatorów.

Etap 1. Opracowanie programów do badań.

Zleceniodawca
Praca statutowa PIAP

Pracę rozpoczęto dnia 10.07.92 zakończono dnia 15.10.92

Kierownik Pracowni
mgr inż. Z. Pietrusiński

Z-ca Dyr. d/s Bad.-Rozwoj.
dr inż. J. Jabłkowski

Kierownik Zespołu
doc. dr inż. J. Korytkowski

Praca zawiera:	Rozdzielnik - ilość egz:
stron 16	Egz. 1
rysunków 1	Egz. 2
fotografii	Egz. 3
tabel	Egz. 4
tablic	Egz. 5
załączników	Egz. 6

Nr rejestr. 6882

1

Analiza deskryptorowa

REGULATORY ELEKTRONICZNE + ALGORYTMY + OPROGRAMOWANIE + ADAPTACJA : BADANIA

Analiza dokumentacyjna

Praca zawiera programy opracowane dla badań symulacyjnych algorytmów autostrojenia regulatorow ciągłych PID. Programy zostały napisane w języku C (do implementacji na komputerze typu PC) i zawierają procedury symulacji: obiektu regulacji, regulatora PID układu autostrojenia oraz programy dla wizualizacji wyników.

Tytuły poprzednich sprawozdań

1. Algorytmy autostrojenia regulatorów i ich implementacja do oprogamowania mikroprocesorowych regulatorów ciałych wolnozmiennych procesów przemysłowych. - Sprawozdanie z pracy PIAP Nr rejestr.6851.

PIAP 41/88 10000

SPIS TREŚCI

3

1. Wstęp

Praca jest kontynuacją tematu "Algorytmy autostrojenia regulatorów i ich implementacja do oprogramowania mikroprocesorowych regulatorów ciągłych wolnozmiennych procesów przemysłowych" w ramach której został wykonany etap 1 p.t. "Przegląd algorytmów autostrojenia oraz wybór i dostosowanie ich do rzeczywistych warunków pracy mikroprocesorowych regulatorów procesów wolnozmiennych."

Obecnie samostrojenie i adaptacja stały się już niemal powszechne w nowych opracowaniach mikroprocesorowych regulatorów procesów wolnozmiennych. Jednak rozwiązania stosowane w poszczególnych aparatach znacznie różnią się w szczegółach, które często decydują o jakości algorytmu, a firmy prowadzą prace nad dalszym ich ulepszeniem.

W wyniku prac prowadzonych w ZAE został zaproponowany oryginalny algorytm oparty w swej idei na zastosowania dwu podstawowych procedur autostrojenia :

- procedurze strojenia zgrubnego opartej na metodzie cyklu granicznego i odpowiedzi skokowej;
- procedurze strojenia dokładnego, opartej na własnych przemyśleniach, obejmującej strojenie regulatora zarówno w wypadku zmieniających się parametrów obiektu jak i zmieniających się zakłóceń oddziaływujących na proces regulacji.

Prace prowadzone w niniejszym etapie miały na celu przygotowanie programów komputerowych, które zostaną wykorzystane w następnym etapie pracy obejmującym badania symulacyjne algorytmów autostrojenia.

2. Ogólna charakterystyka programów

Główne zadania realizowane przez przygotowane oprogramowanie to :
- symulacja obiektu regulacji i zakłóceń oddziaływujących na niego,
- algorytm regulatora ciągłego PID o przestrajanych nastawach,
- realizacja funkcji autostrojenia,
- prezentacja wyników.

Obiekt regulacji, zgodnie z wcześniejszymi założeniami posiada transmitancję typu

$$\frac{k_{ob}}{1 + sT_F} e^{-s\tau}$$

Przy czym przyjęto, że $k_{ob} = 1$. Nie zmniejsza to ogólności badań, gdyż zachowanie się zamkniętego układu regulacji zależy od iloczynu wzmocnienia obiektu i regulatora, a wzmocnienie regulatora jest nastawialne.

W celu umożliwienia przeprowadzenia badań dodatkowych, w programach przewidziano również obiekt dwuinercyjny o transmitancji typu

$$\frac{k_{ob}}{(1 + sT_1)(1 + sT_2)} e^{-s\tau}$$

Program algorytmu regulatora ciągłego PID umożliwia nastawę, a także przestrajanie następujących parametrów :
- wzmocnienia regulatora K_P,
- stałej czasowej całkowania T_I,
- stałej czasowej różniczkowania T_D,
- stałej czasowej T_{FP}.

Przestrajanie (autostrojenie) nastaw regulatora odbywa się z nastawianym współczynnikiem wagi oddzielnie dla każdego parametru regulacyjnego. Stała czasowa filtracji sygnału różniczkowania nie podlega autostrojeniu.

Przyrostowe, procentowe zmiany nastaw parametrów dynamicznych wyznaczone są przez procedurę strojenia dokładnego. Program

algorytmu autostrojenia dokładnego został zrealizowany w oparciu o szeregowe połączenie czterech członów całkujących, o tej samej stałej czasowej całkowania.

Schemat strukturalny układu autostrojenia i jego wyjściowe parametry pokazane są na rys.1. W trakcie badań należy się liczyć z ewentualnymi korektami układu, które mogą obejmować zarówno zmianę wartości wstępnie zaproponowanych współczynników jak i pewne zmiany strukturalne.

Programy prezentacji wyników obejmują :
- zapis wyników symulacji do zbioru dyskowego,
- prezentację wyników w formie wykresu.

Program główny realizuje funkcję regulacji z zadanym krokiem próbkowania i z zadaną ilością próbek.

Zapis wyników na dysku umożliwia rejestrację wyników w długim okresie czasu bez obawy przed przepełnieniem pamięci operacyjnej. Zapamiętywane są przebiegi :
- sygnału wejściowego,
- odchyłki regulacji,
- sygnału regulowanego (wyjścia obiektu regulacji),
- sygnału wyjściowego regulatora,
- sygnałów strojenia.

Program analizuje wartości zmiennych, zapamiętuje wartość maksymalną i sam określa skalę na osi pionowej. Skala czasu ustalana jest przez operatora.

Istnieje możliwość prezentacji wybranych przebiegów zgodnie z deklaracją operatora. Jest możliwość zwiększania lub zmniejszania wybranych przebiegów lub ich fragmentów.

3. Wydruki programów

Programy zostały napisane w języku C, co stwarza możliwość stosunkowo prostej implementacji algorytmów w regulatorach zrealizowanych w oparciu o różne typy mikroprocesorów. Programy opatrzone są bogatymi komentarzami zapewniającymi przejrzystość i ułatwiającymi wprowadzanie ewentualnych poprawek.

W czasie opracowywania programów podlegały one wstępnemu testowaniu w związku z czym wproadzono w nich szereg poprawek w stosunku do wersji początkowej.
Zamieszczona wersja jest już z kolei piątą wersją oprogramowania. Ze wzgledu na konieczność zapewnienia odpowiedniej dokładności większość obliczeń wykonywana jest w arytmetyce z podwójną precyzją.

```
/***********************************************************************/
/*** Piata wersja programu symulacyjnego obiektu regulacj typu      ***/
/*** opoznienie z inercja z regulatorem PID. Wyniki symulacji za-   ***/
/*** pisywane sa do zbioru. Obliczenia wykonywane sa na zmiennych   ***/
/*** typu double float. Po zakonczeniu eksperymentu wywolywany      ***/
/***  jest program prezentowania symulacji na wykresie. Poprawiony  ***/
/*** zostal algorytm PID (filtracja skladowej D)                    ***/
/*** 17 wrzesnia 1992 ***************************************************/
/***********************************************************************/

#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<io.h>
#include<string.h>

/*=  ================================================================*/
                        /* parametry eksperymentu */
#define GLOBAL_TIME     2000    /* globalny czas eksperymentu */
#define TAU     1.0             /* okres probkowania */
#define ZAKRES_MAX      10000.0
#define ZAKRES_MIN      -10000.0
                        /* parametry sygnalu wejsciowego */
#define CZEKAJ  50              /* opoznienie */
#define X_START 0.0             /* wartosc poczatkowa */
#define SKOK    5000.0          /* wartosc skoku */
                        /* nastawy regulatora PID */
#define KP              0.5
#define TI              52.0
#define TD              20.0
#define TFP             2.0

                        /* parametry obiektu opoznienie z inercja */
#define TF      100.0           /* stala czasowa inercji */
#define DELAY   40              /* czas opoznienia */
                        /* parametry obiektu dwuinercyjnego */
#define T1              5.0
#define T2              6.0

                        /* parametry czlonow calkujacych */
#define TII             0.55*((double)DELAY*TAU)
#define STALA   10000.0
/*=  ================================================================*/
                        /* tablice wynikowe */
double ypop,a[DELAY];           /* wyniki chronione obiektu1 */
double sum,wejpop,delta_pop;    /* wyniki chronione regulatora */
double kp,ti,td;
double yp1,yp2;                 /* obiektu dwuinercyjnego */
                                /* podprogramu adaptacyjnego */
double se1,se2,se3,se4,sum_e1,sum_e2,sum_e3,sum_e4;
double in_p1,in_p2;

double wyl_kp;

double obiekt_1();
double pid();
double obiekt_2();
double calka();

#define ON      1               /* zbior otwarty do zapisu */
#define OFF 0                   /* zbior nie otworzony */
#define NAZWA   20              /* Max. dlugosc nazwy zbioru */
```

```
int arg_c;
char *arg_v[NAZWA];
char name[NAZWA];
int dp,dd;
FILE *out;

void main(argc,argv)
        int argc;
        char *argv[];
{
        char c;                 /* pomocniczy klawisz */
        int opened = OFF;       /* zbior nie otworzony */

        int i,t,tp,j,jj;
        int cg,cy,cb,cr;        /* wlaczanie/wylaczanie kolorowych ch-k */
        double wz,e,yo,y;
        double wzp,ep,yop,yp;
        long int max,may,mg,my,mb,mr;   /*wzmocnienia poszczegolnych colorow*/
        int g_driver,g_mode,max_x,max_y;
        char tekst[5];
        long int kursor = 0L;
        kp = KP; ti = TI; td = TD;
        cg = cy = cb = cr = 0;


/*************************************/
        wz = X_START;
        y = e = yo = yp = ep = yop = 0.0;
        ypop = 0.0;
        yp1 = yp2 = 0.0;
        wejpop = 0.0;
        sum = 0.0;
        delta_pop = 0.0;

        se1=se2=se3=se4=sum_e1=sum_e2=sum_e3=sum_e4=0.0;
        in_p1 = in_p2 = 0.0;
        wyl_kp = 0.0;
        t = tp = 0;

        for(i=0; i<=DELAY-1; i++)
                a[i] = 0.0;

/* zbior juz otwarty, ustawianie kursora na koniec */
        if(opened == ON)
                fseek(out,kursor,SEEK_END);
/* pierwsze otwieranie zbioru do zapisu */
        if(opened == OFF)
        {
          arg_c = argc;
          if(argc > 1 )
                strcpy(arg_v[1],argv[1]);
          if(otworz_zapis() == 1)
                printf("Nie mozna otworzyc zbioru do zapisu !!!");
          opened = ON;
        }

/* petla oczekiwania na start eksperymentu */
        for(i=0; i<=CZEKAJ; i++,t++)
        {
                zapisz(t,wz,e,y,yo);
/*              print_wynik(t,wz,e,y,yo);*/
        }
```

```
/* rozpoczecie eksperymentu, ustawienie wartosci poczatkowych */

        wz = X_START + SKOK;

                if(wz > ZAKRES_MAX)
                        wz = ZAKRES_MAX;
                if(wz < ZAKRES_MIN)
                        wz = ZAKRES_MIN;

/* glowna petla obliczeniowa */
        for(i=0; t<=GLOBAL_TIME; i++,t++)
        {

/* petla zamknieta regulacji z obiektem opoznienie z inercja */
                e = wz - yo;
                if(e > ZAKRES_MAX)
                        e = ZAKRES_MAX;
                if(e < ZAKRES_MIN)
                        e = ZAKRES_MIN;

                y = calka(e)*STALA;

/*              kp = KP + wyl_kp;*/

                yo = obiekt_1(pid(e));
        /* proba obiektu obiekt_1() */
/*              yo = obiekt_1(wz);*/
/*              yo = pid(wz);*/
/*              e = y = 0.0;*/


        /* odpowiedzi obiektow na skok */
/*      y = obiekt_1(wz);               */
/*      yo = obiekt_2(wz);              */
/*       e = 0;                                 */

                zapisz(t,wz,e,y,yo);
/*              print_wynik(t,wz,e,y,yo);*/
        }
/* zapis parametrow eksperymentu */
fprintf(out,"Nastawy regulatora:\n\tKp = %3.2f,\tTi = %3.2f,\tTd = %3.2f (Tf= %3.2f)\n"
,TFP);
fprintf(out,"Parametry obiektu:\n\tTopz = %d,\tTf = %3.2f\n",DELAY,TF);
fprintf(out,"Parametry sygnalow:\n\tWZ = %5.2f,\tXz = %5.2f\n",X_START,SKOK);

/*      zamknij_zapis();*/

/*      otworz_odczyt();*/


/* zakonczenie zapisu eksperymentu i wizualizacja */
        /* ustawienie dysku ze zbiorami .bgi */
        setdisk(dp);
        detectgraph(&g_driver,&g_mode);
        if(g_mode == EGAHI)
                g_mode = EGALO;
        if(g_driver == CGA)
                g_mode = CGAHI;
        initgraph(&g_driver,&g_mode,"");
        setfillstyle(0,0);
        settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
        setusercharsize(1,1,3,1);
        bar(0,0,max_x=getmaxx(),max_y=(getmaxy()-10));
```

```
        fseek(out,kursor,SEEK_SET);

/* wyszukanie maksimum */



        max = 0;
        for(i=0; i<=GLOBAL_TIME; i++)
        {
                fscanf(out,"%d%lf%lf%lf%lf",&t,&wz,&e,&y,&yo);
                if(max <= (int)fabs(wz))
                        max = (int)fabs(wz);
                if(max <= (int)fabs(e))
                        max = (int)fabs(e);
                if(max <= (int)fabs(y))
                        max = (int)fabs(y);
                if(max <= (int)fabs(yo))
                        max = (int)fabs(yo);
        }
        max = max*2;
        mg = mb = my = mr = may = max;
        j = 1;
        fseek(out,kursor,SEEK_SET);
        c = NULL;
        do
        {

                clearviewport();
                setcolor(WHITE);
/* rysowanie osi x,y */
                setlinestyle(SOLID_LINE,0XFFFF,NORM_WIDTH);
                line(0,0,0,max_y);
                line(0,max_y/2,max_x,max_y/2);

                if(g_driver == EGA || g_driver == VGA)
                        setlinestyle(SOLID_LINE,0XFFFF,THICK_WIDTH);
/*      outtextxy(max_x-40,max_y/2+10,"time");*/
outtextxy(0,max_y+3,"-/+ Rozszerzenie/skracanie skali czasu; G,Y,B,R kolory; W,Z wzmocn
end;");
/* rysowanie wykresu */
                if(c == 'q' || c == 'Q')
                        break;
                if(c == '+')
                        j--;
                if(c == '-')
                        j++;
                if(c == 'y' || c == 'Y')
                        cy = !cy;
                if(c == 'g' || c == 'G')
                        cg = !cg;
                if(c == 'b' || c == 'B')
                        cb = !cb;
                if(c == 'r' || c == 'R')
                        cr = !cr;
                if(c == 'w' || c == 'W')
                {
                        if(cy == 0 && my > may/10)
                                my -= may/10;
```


11

```
                if(cg == 0 && mg > may/10)
                        mg -= may/10;
                if(cb == 0 && mb > may/10)
                        mb -= may/10;
                if(cr == 0 && mr > may/10)
                        mr -= may/10;
        }
        if(c == 'z' || c == 'Z')
        {
                if(cy == 0)
                        my += may/10;
                if(cg == 0)
                        mg += may/10;
                if(cb == 0)
                        mb += may/10;
                if(cr == 0)
                        mr += may/10;
        }
        if(c == 'n' || c == 'N')
                ;
        else
        {
                kursor = 0L;
        }
        fseek(out,kursor,SEEK_SET);
        fscanf(out,"%d%lf%lf%lf%lf",&tp,&wzp,&ep,&yp,&yop);
                for(i = 0; i <= max_x; i++)
        {
                if(j >= 1)
                {
                  for(jj=1; jj<=j; jj++)
if(fscanf(out,"%d%lf%lf%lf%lf",&t,&wz,&e,&y,&yo) == EOF || t >= GLOBAL_TIME)
                        goto the_end;
                }
                else
                {
                        jj = abs(j)+2;
                        if(i%jj == 0)
if(fscanf(out,"%d%lf%lf%lf%lf",&t,&wz,&e,&y,&yo) == EOF || t >= GLOBAL_TIME)
                        goto the_end;
                }
                setcolor(WHITE);
                sprintf(tekst,"%d",t);
                if(i == max_x/4)
                        outtextxy(max_x/4,max_y/2+10,tekst);
                if(i == max_x/2)
                        outtextxy(max_x/2,max_y/2+10,tekst);
                if(i == max_x*3/4)
                        outtextxy(max_x*3/4,max_y/2+10,tekst);
                if(i == max_x)
                        outtextxy(max_x-40,max_y/2+10,tekst);
                sprintf(tekst,"+%d",max/2);
                outtextxy(0,0,tekst);
                sprintf(tekst,"+%d",max/4);
                outtextxy(0,max_y/4,tekst);
                sprintf(tekst,"-%d",max/2);
                outtextxy(0,max_y-5,tekst);
                sprintf(tekst,"-%d",max/4);
                outtextxy(0,max_y*3/4,tekst);
                if(t >= GLOBAL_TIME-1)
                        break;
                        if(cr == 0)
```

```
                {
                        if(g_driver == EGA || g_driver == VGA)
                                setcolor(RED);
                        else
                                setlinestyle(DOTTED_LINE,0xFFFF,NORM_WIDTH);
line(i,(max_y/2-((long int)ep*max_y)/mr),i,(max_y/2-((long int)e*max_y)/mr));
                }
                if(cg == 0)
                {
                        if(g_driver == EGA || g_driver == VGA)
                                setcolor(GREEN);
                        else
                                setlinestyle(CENTER_LINE,0xFFFF,NORM_WIDTH);
line(i,(max_y/2-((long int)yp*max_y)/mg),i,(max_y/2-((long int)y*max_y)/mg));
                }
                if(cy == 0)
                {
                        if(g_driver == EGA || g_driver == VGA)
                                setcolor(YELLOW);
                        else
                                setlinestyle(DASHED_LINE,0xFFFF,NORM_WIDTH);
line(i,(max_y/2-((long int)wzp*max_y)/my),i,(max_y/2-((long int)wz*max_y)/my));
                }
                        if(cb == 0)
                {
                        if(g_driver == EGA || g_driver == VGA)
                                setcolor(LIGHTBLUE);
                        else
                                setlinestyle(DOTTED_LINE,0xFFFF,NORM_WIDTH);
line(i,(max_y/2-((long int)yop*max_y)/mb),i,(max_y/2-((long int)yo*max_y)/mb));
                }
the_end:
                tp = t;wzp = wz;ep = e;yp = y;yop = yo;

                }
        kursor = ftell(out);

        }
        while((c=getch()) == '+' || c == '-' || c != 'q' || c != 'Q');
        clearviewport();
        closegraph();
/*************************************/
        setdisk(dd);
        zamknij_odczyt();

        exit(0);
}

/*************************************************************************/

/* algorytm obiektu typu opoznienie z inercja */
double obiekt_1(xf)
         double xf;
{
        extern double ypop;
        extern double a[];
        int p;
        double yf;
        double tautf = TAU+TF;

                for(p=0; p<=DELAY-1; p++)
                        a[p] = a[p+1];
```


13

```
                a[DELAY-1] = xf;

                yf = (a[0]*TAU+ypop*TF)/(tautf);
                if(yf > ZAKRES_MAX)
                        yf = ZAKRES_MAX;
                if(yf < ZAKRES_MIN)
                        yf = ZAKRES_MIN;

                ypop = yf;
        return(yf);
}
/* algorytm obiektu typu dwuinercyjnego */
double obiekt_2(xf)
         double xf;
{
extern double yp1,yp2;
double t1t,t2t,t12,tt;
double yf;

        t1t = T1*TAU;
        t2t = T2*TAU;
        t12 = T1*T2;
        tt = TAU*TAU;

        yf=( (xf*tt)+(yp1*(t1t+t2t+2*t12))-(yp2*t12) )/(tt+t1t+t2t+t12);
                if(yf > ZAKRES_MAX)
                        yf = ZAKRES_MAX;
                if(yf < ZAKRES_MIN)
                        yf = ZAKRES_MIN;

                yp2 = yp1;
                yp1 = yf;
                return(yf);
}
/*************************************************************************/

double pid(wej)
        double wej;
{

double wyj;
double delta_wej=0.0;
extern double sum,wejpop,delta_pop;
extern double kp,ti,td;
double tautfp = (TFP+TAU);
double tfp = TFP;

/*      sum = sum + wejpop;*/
        delta_wej = ((wej-wejpop)*TAU + delta_pop*tfp)/tautfp;


        wyj = kp*wej + (kp*sum*TAU) /ti + (kp*td/TAU)*delta_wej;
                if(wyj > ZAKRES_MAX)
                        wyj = ZAKRES_MAX;
                if(wyj < ZAKRES_MIN)
                        wyj = ZAKRES_MIN;


        sum = sum + wejpop;
        wejpop = wej;
        delta_pop = delta_wej;
```


14

```
return(wyj);
}
/*****************************************************************/
double calka(in)
        double in;
{
        extern double se1,se2,se3,se4,sum_e1,sum_e2,sum_e3,sum_e4;
        extern double in_p1,in_p2;
        extern double kp,ti,td;
        extern double wyl_kp;
        double ek = 0.0,tii;
        double delta_kp,delta ti,delta td;
        double taut = 0.16*TII + TAU;
        double tli = 0.16*TII;

        tii = TII;

        in_p1 = (in*TAU + in_p1*tli)/taut;
        in_p2 = (in_p1*TAU + in_p2*tli)/taut;

        ek = (in_p2/STALA)-1.5*se1-3.250*se2-1.9*se3-se4;

        sum_e1 = sum_e1 + ek;
        se1 = (TAU*sum_e1)/tii;

        sum_e2 = sum_e2 + se1;
        se2 = (TAU*sum_e2)/tii;

        sum_e3 = sum_e3 + se2;
        se3 = (TAU*sum_e3 )/tii;

        sum_e4 = sum_e4 + se3;
        se4 = (TAU*sum_e4 )/tii;


delta_kp = 1.0*se4*se4 - 1.1*se3*se3 + 1.65*se2*se2 - 2.5*se1*se1 + 0.41*ek*ek;
        wyl_kp = wyl_kp + 0.001*delta_kp;

/*      printf("delta_kp = %lf\t\t wyl_kp = %lf\n",delta_kp,wyl_kp);*/
/*      getch();*/

        return(wyl_kp);

}


/*****************************************************************/
print_wynik(time,in1,in2,out1,out2)
        int time;
        double in1,in2,out1,out2;
{
printf("%4d   %5.2f   %5.2f     %5.2f   %5.2f\n",time,in1,in2,out1,out2);
        if(time%20 == 0)
                getch();
}

/* otwieranie zbioru do zapisu                  */

otworz_zapis()
{
extern int arg_c;
```

```
extern char *arg_v[];
extern dp,dd;
extern char name[];
char *roz = ".TAB";
char *pkt;
extern FILE *out;
int i;
                /* WYBOR DYSKU */
        dp = getdisk();
        dd = setdisk();
        printf(" Wybierz dysk: ");
        for(i='A'; i<=('A'-1+dd); i++)
                printf(" %c,",i);
        printf("...?\n");
        if((dd=toupper(getch())) >= 'A' && dd <='F')
                setdisk(dd-='A');
        else
                dd = dp;
                /* nazwa zbioru do zapisu */
                if(arg_c > 1)
                        strcpy(name,arg_v[1]);
                else
                {
                        printf(" Podaj nazwe zbioru ... ");
                        scanf("%12s",&name);
                }
                strupr(name);
                strcpy(arg_v[1],name);
                if((pkt=strchr(name,'.')) == NULL)
                        strcat(name,roz);
                else
                        strcpy(pkt,roz);
                strcpy(arg_v[1],name);
                if((out=fopen(name,"wt+")) == NULL)
                        return(1);
                printf(" Zapis do zbioru %c: \\%s\n",dd+'A',name);
}
/* ZAMKNIECIE ZAPISU DO ZBIORU  */
zamknij_zapis()
{
extern int dp;
extern FILE *out;
        fclose(out);
/*      setdisk(dp);*/
}
/* OTWORZENIE ZBIORU DO ODCZYTU */
otworz_odczyt()
{
extern char name[];
extern int dp;
extern FILE *out;
        if((out=fopen(name,"r")) == NULL)
                return(1);
        printf(" Odczyt ze zbioru %c: \\%s\n",dp,name);
}
/* ZAMYKANIE CALOSCI ZBIOROW */
zamknij_odczyt()
{
extern int dp;
extern FILE *out;
        fclose(out);
        setdisk(dp);
```

```
}
zapisz(time,lx1,lx2,lx3,lx4)
        int time;
        double lx1,lx2,lx3,lx4;
{
        extern FILE *out;
        fprintf(out,"%d\t%5.5f\t%5.5f\t%5.5f\t%5.5f\n",time,lx1,lx2,lx3,lx4);
}
```


17