ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1990
Seria: TRANSPORT z. 16 Nr kol. 1100

Stanisław KRAWIEC

REALIZACJA STRUKTUR SYMULACYJNYCH W JĘZYKU TURBO PASCAL DLA POTRZEB MODELU SYMULACYJNEGO RUCHU POCIĄGÓW

Streszczenie. W artykule przedstawiono realizację struktury symulacyjnej w języku Turbo PASCAL v.5.0. Przedstawiono zasady tworzenia modułu SIMULATION, modułu SIMPROCESS, i=1...LPROCESÓW dla ww. struktury oraz zasady komunikacji między obiektami tworzonymi podczas realizacji procesu symulacyjnego. Jako przykład wykorzystania zrealizowanej struktury symulacyjnej podano strukturę modelu symulacyjnego do symulacji ruchu pociągów na linii Kolejowego Ruchu Regionalnego.

1. Wprowadzenie

Zespół pracowników Instytutu Transportu Politechniki Śląskiej zrealizował w latach 1986-1989 model symulacyjny ruchu pociągów na sieci kolejowej dla potrzeb regulacji ruchu [4]. Realizacja kolejnych etapów tych prac, prowadzonych w ramach programu badawczego RP.I.09, doprowadziła do stworzenia dwóch zasadniczych modułów oprogramowania symulacyjnego:

- modułu MODEL, symulującego ruch pociągów na dowolnym fragmencie sieci kolejowej [1,2,3,4,5];
- modułu MAPA, odwzorowującego w posób dynamiczny ruch pociągów na symulowanym fragmencie sieci kolejowej [15,16,17,18].

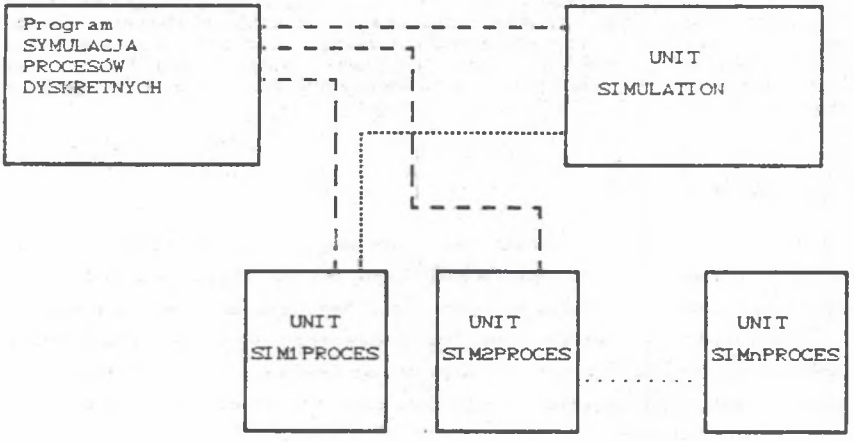
Całość oprogramowania zrealizowano w języku LOGLAN w ramach prac poświęconych testowaniu tego języka na mikrokomputerze IBM PC, zrealizowanego przez zespółpracowników Instytutu Informatyki Uniwersytetu Warszawskiego pod kierownictwem prof. A.Salwickiego. W 1989 roku zrealizowano wariant dwukomputerowy [17], umożliwiający współdziałanie modułu MODEL oraz MAPA. Ze względu na ograniczenia interpretera języka LOGLAN nie udało sie zintegrowanie całego oprogramowania w ramach wariantu jednokomputerowego. Ze względu na brak efektywnej implementacji kompilatora LOGLAN-u do realizacji zintegrowanego modelu ruchu pociągów na przykładzie nowo projektowanej linii KRR postanowiono wykorzystać zasoby języka Turbo PASCAL v.5.0, zachowując jednak "loglanowski" styl programowania obiektowego.

W artykule poprzednim niniejszego zeszytu [19] przedstawiono

możliwości realizacji odpowiednika loglanowskiego modułu SIMULATION na bazie języka Turbo PASCAL, co łącznie ze standardowymi możliwościami wersji 5.0 tego języka umożliwiło stworzenie struktury symulacyjnej, umożliwiającej stosowanie z istoty i filozofii metod programowania obiektowego.

2. Struktura programu symulacyjnego w języku Turbo PASCAL

Ogólna struktura programu symulacyjnego w języku Turbo PASCAL przedstawiona została w [19]. Poniżej przedstawiono strukturę programu symulacyjnego, będącą podstawową wersją budowy dla realizacji testowego programu symulacyjnego, a następnie modelu symulacyjnego ruchu pociągów.



Rys.1. Bazowa struktura programu symulacyjnego
Fig.1. A base structure of simulation programm

Program główny, czyli program SYMULACJA_PROCESÓW_DYSKRETNYCH spełnia następujące funkcje:
- wiąże wszystkie moduły występujące w modelu symulacyjnym,
- kończy wykonywanie programu symulacyjnego.

Bazowa struktura tego programu może być następująca:

```
PROGRAM SYMULACJA_PROCESÓW_DYSKRETNYCH;
USES SIMULATION, SIM1PROCES, SIM2PROCES,...,SIMnPROCES;
BEGIN
```

```
PROCES;
{zainicjowanie symulacji poprzez wywołanie procedury PROCES z UNIT-u
SIMULATION}
WRITELN('KONIEC SYMULACJI_PROCESÓW_DYSKRETNYCH');
END. {PROGRAM SYMULACJA_PROCESÓW_DYSKRETNYCH}
```

UNIT SIMULATION spełnia rolę loglanowskiego modułu SIMULATION. Struktura bazowa tego modułu może być przedstawiona w sposób poglądowy, choć nieformalny następująco:

```
UNIT SIMULATION;
INTERFACE
CONST
 MAXYMALNA_LICZBA_PROCESÓW=     {szacowana z góry np.100}
 CZAS_AKTUALNY: LONGINT=        {czas początkowy np.0}
 IND: INTEGER = 0               {wskaźnik procesu aktywnego}
 POPZĄDKOWANIE_TABLICY_PRCESÓW= FALSE;
TYPE
 TYP_NAZWY_PROCESU = (PROCES_1,PROCES_2,...,PROCES_N);
 {ilość nazw  procesów dla konkretnego modelu jest znana}
 TABLICA_PROCESÓW = ARRAY[1..MAXYMALNA_LICZBA_PROCESÓW] OF
                    RECORD
                     TYP                      :TYP_NAZWY_PROCESU;
                     P                        :POINTER;
                     PLANOWY_CZAS_AKTYWACJI:LONGINT;
                     CZY_USUNĄĆ               :BOOLEAN;
                     NUMER_OBIEKTU_DANEGO_TYPU:INTEGER;
                    END;
VAR
 MAXYMALNY_PROCES_WYGENEROWANY    :1..MAXYMALNA_LICZBA_PROCESÓW;
 PROCESY                          :TABLICA_PROCESÓW;
 CZAS_ZAKOŃCZENIA_SYMULACJI       :LONGINT;
PROCEDURE PROCES;
FUNCTION F(TYP_PROCESU : TYP_NAZWY_PROCESU;
           NUMER_PROCESU : INTEGER):INTEGER;

IMPLEMENTATION
 USES SIMULATION, SIM1PROCES, SIM2PROCES,...,SIMnPROCES;
 FUNCTION F(TYP_PROCESU : TYP_NAZWY_PROCESU;
            NUMER_PROCESU : INTEGER):INTEGER;
 BEGIN
  {funkcja odszukuje aktualny numer procesu w tablicy procesów}
 END;
```

```
PROCEDURE PROCES;
 {- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
 PROCEDURE PORZĄDKOWANIE;
 VAR I:INTEGER;
 BEGIN
  FOR I:=1 TO MAXYMALNY_PROCES_WYGENEROWANY DP
   IF PROCESY[I].CZY_USUNĄĆ
      THEN {przesuwanie procesu o numerze aktualnie najwyższym na miejsce
            procesu usuniętego z procesu symulacyjnego za pomocą
            zlecenia DISPOSE};
  MAXYMALNY_PROCES_WYGENEROWANY:=MAXYMALNY_PROCES_WYGENEROWANY-1;
  PORZĄDKOWANIE_TABLICY_PROCESÓW:=FALSE;
 END; {PROCEDURA PORZĄDKOWANIE}
 {- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
 PROCEDURE WYBÓR_PROCESU;
 VAR I,MINIMALNY_CZAS:INTEGER;
 BEGIN
  MINIMALNY_CZAS:=PROCESY[1].PLANOWY_CZAS_AKTYWACJI;
  I:=1;
  IND:=1;
  WHILE I<MAXYMALNY_PROCES_WYGENEROWANY DO
   BEGIN
    I:=I+1;
    IF PROCESY[I].PLANOWY_CZAS_AKTYWACJI < MINIMALNY_CZAS
       THEN BEGIN
              MINIMALNY_CZAS:=PROCESY[I].PLANOWY_CZAS_AKTYWACJI;
              IND:=I;
             END;
   END; {wybrano proces IND}
  IF MINIMALNY_CZAS>=0 THEN BEGIN
                             CZAS_AKTUALNY:=CZAS_AKTUALNY+MINIMALNY_CZAS;
                             FOR I:=1 TO MAXYMALNY_PROCES_WYGENEROWANY DO
                              PROCESY[I].PLANOWY_CZAS_AKTYWACJI:=
                              PROCESY[I].PLANOWY_CZAS_AKTYWACJI.MINIMALNY
                              _CZAS;
                            END;
  CASE PROCESY[IND].TYP OF
   PROCES_1: PROCES1(PROCESY[IND].P);
   PROCES_2: PROCES2(PROCESY[IND].P);
   :
   :
    PROCES_N: PROCESN(PROCESY[IND].P);
  END; {CASE}.
```

```
END; {PROCEDURA WYBÓR_PROCESU}
{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
BEGIN
 {generowanie procesów}
 ......
 {przykładowe wygenerowanie 5 obiektu procesu typu PROCES_i, będącego
 np. 13 obiektem wygenerowanym w modelu}
 NEW(Pi);Pi^.N:=5;
          Pi^.E:=1;
          Pi^.atrybut_merytoryczny_procesu:=
 PROCESY[13].TYP:=PROCES_i;
 PROCESY[13].PLANOWY_CZAS_AKTYWACJI:=
 PROCESY[13].P:=Pi;
 PROCESY[13].CZY_USUNĄĆ:=FALSE;
 PROCESY[13].NUMER_OBIEKTU_DANEGO_TYPU:=5;
{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
 AKTUALNA_LICZBA_WYGENEROWANYCH_OBIEKTÓW_TYPU_i:=5;
 MAXYMALNY_PROCES_WYGENEROWANY:=13;
{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
{wybór procesu}
 WHILE CZAS_AKTUALNY< CZAS_ZAKOŃCZENIA_SYMULACJI DO
  BEGIN
   IF PORZĄDKOWANIE_TABLICY_PROCESÓW THEN PORZĄDKOWANIE;
   WYBÓR_PROCESU;
  END;
 EXIT {do programu głównego}
END;
BEGIN
 WRITELN('PODAJ CZAS ZAKOŃCZENIA SYMULACJI');
 READLN(CZAS_ZAKOŃCZENIA_SYMULACJI);
END. {PROCES}
```

Do realizacji osi czasu wybrano wariant tymczasowy, najprostszy, polegający na każdorazowym przeglądaniu przy wyborze procesu atrybutu PLANOWY_CZAS_AKTYWACJI wszystkich obiektów. Procedura PORZĄDKOWANIE używana jest każdorazowo po użyciu instrukcji DISPOSE. Generowanie procesów (początkowe) nie musi być realizowane w module SIMULATION. Może być realizowany w oddzielnym module, np. module DIALOG, który musiałby być wywołany z programu głównego przed wywołaniem procedury PROCES. Funkcja F jest podstawową funkcją umożliwiającą realizację loglanowskich instrukcji operujących na atrybutach czasu (HOLD, CONTINUE itd.).

Bazowa struktura modułu SIMIPROCES może być przedstawiona następująco (ponownie nieformalnie dla celów poglądowych):

```
UNIT SIMiPROCES; {i=1,...,N}
{moduł PROCESi}
INTERFACE
 TYPE
  REKORD_STANU_PROCESUi_=^REKORD_STANU_PROCESUi;
  REKORD_STANU_PROCESUi=RECORD
                          E:BYTE;    {etykieta kontynuacji}
                          N:INTEGER; {numer obiektu typu i}
                          :
                          atrybut_merytoryczny_procesu:...
                          :
                        END;
 VAR
  Pi:REKORD_STANU_PROCESUi_;
  AKTUALNA_LICZBA_WYGENEROWANYCH_OBIEKTÓW_TYPU_i:INTEGER;
 PROCEDURE PROCESi(W:REKORD_STANU_PROCESU);
IMPLEMENTATION
USES SIM1PROCES,...,SIMi-1PROCES,SIMi+1PROCES,...,SIMnPROCES,SIMULATION;
 {- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
 PROCEDURE PROCESi(W:REKORD_STANU_PROCESU);
 CONST ET1_=1;ET2_=2,...,ETk_=k;
 LABEL ET1,ET2,...,ETk {etykiety kontynuacji}
  BEGIN
   WITH W^ DO
    BEGIN
     CASE E OF {wymóg techniczny w PASCAL-u}
      ET1_:GOTO ET1;
      ET2_:GOTO ET2;
      :
      ETk_:GOTO ETk;
      ELSE BEGIN WRITELN('BŁĄD'); HALT; END;
     END; {CASE}
     ET1:{...}
         {początkowe akcje procesu}
         {...}
         E:=ET2_;EXIT;
     ET2:{...}
         {dalsze akcje procesu};
         {...}
         E:=ET3_;EXIT;
     ET3:{...}
         {dalsze akcje procesu};
         {...}
```

```
        E:=ETk_;EXIT;
        :
    ETk:(...)
        (dalsze akcje procesu);
        (...)
        E:=ET1_;EXIT;
  END;
END; (PROCESi)
(- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -)
BEGIN
END. (SIMiPROCES)
```

Każdy moduł typu SIMiPROCES (i=1,...,N) w modelu symulacyjnym jest tylko pewnym wzorcem obiektu danego typu. Właściwymi obiektami w modelu są rekordu stanów zadeklarowane w INTERFACE-ie tych modułów, a generowane w dowolnym miejscu programu zleceniem NEW(Pi). Wszelkie operacje na rekordach stanu, czyli na obiektach modelu symulacyjnego, przeprowadzane są w ramach akcji procesu między kolejnymi etykietami modułów typu SIMiPROCES.

W celu umożliwienia przeprowadzenia operacji na obiekcie, czyli na konkretnym rekordzie stanu, musi być znany aktualny numer obiektu, pod którym jest identyfikowany w tablicy PROCESY (numer ten nie jest stały w procesie symulacyjnym). Znajomość tego numeru umożliwia:

- dostęp do zmiennej P, typu POINTER, wskazującej adres tego obiektu w pamięci,
- dostęp do atrybutu czasu obiektu (PLANOWY_CZAS_AKTYWACJI), umożliwiający sterowanie modelem.

Aktualny numer dowolnego obiektu w tablicy PROCESY udostępnia funkcja F, której argumentami są: typ procesu i numer obiektu danego typu (np. F(POCIĄG,6)). Numer obiektu aktualnie aktywnego udostępnia zmienna IND. Umożliwia to proste zrealizowanie odpowiedników loglanowskich funkcji HOLD, PASSIVATE, SCHEDULE, CANCEL, RUN. Przykładowo odpowiednikiem loglanowskiej instrukcji CALL HOLD(100) jest instrukcja: PROCESY[IND].PLANOWY_CZAS_ _AKTYWACJI:=PROCESY[IND].PLANOWY_CZAS_AKTYWACJI+100, natomiast odpowiednikiem instrukcji CALL SCHEDULE(POCIĄG(6),70); jest instrukcja: PROCESY(F(POCIĄG,6)].PLANOWY_CZAS_AKTYWACJI:=PROCESY[F(POCIĄG,6)].PLANOWY_ _CZAS_AKTYWACJI+70.

3. Struktura modelu symulacyjnego ruchu pociągów w języku Turbo PASCAL

Dla realizacji zintegrowanego modelu symulacyjnego przyjęto i zrealizowano program symulacyjny o następującej strukturze:

```
PROGRAM MODELMAIN;

USES BAZA_KRR,DIALOG,SIMULATION,SIM1PROCES,SIM2PROCES,MAPA,MAPA1,MAPA2,POMOC;
BEGIN
PROCES;
WRITELN('KONIEC SYMULACJI');
SETKURSOR;
END. { PROGRAM MODELMAIN }

UNIT SIMULATION;
{ Modul PROCES }

INTERFACE

CONST

   MAX_L_PROCESOW=100;
   CZAS_AKTUALNY:LONGINT=0;
   IND :INTEGER=0;
   PORZADKOWANIE_TABLICY_PROCESOW:BOOLEAN=FALSE;

TYPE

   TYP_NAZWY_PROCESU = (POCIAG,STEROWANIE,ZWOLNIENIE_PRZEBIEGU);
   TABLICA_PROCESOW  = ARRAY[1..MAX_L_PROCESOW] OF
                       RECORD
                        TYP                 : TYP_NAZWY_PROCESU;
                        P                   : POINTER;
                        PLANOWY_CZAS_AKTYW  : INTEGER;
                        CZY_USUNAC          : BOOLEAN;
                        NR_OBJ_DANEGO_TYPU  : INTEGER;
                       END;

VAR

   MAX_PROCES_WYGENEROWANY    : 1..MAX_L_PROCESOW;
   PROCESY                    : TABLICA_PROCESOW;
   CZAS_ZAKONCZENIA_SYMULACJI : LONGINT;
   MAX_L_POCIAGOW             : BYTE;

PROCEDURE PROCES;

FUNCTION FCTYP_PROCESU   : TYP_NAZWY_PROCESU;
          NUMER_PROCESU : INTEGER) : INTEGER;

FUNCTION XTIME ( XX : STRING ) : LONGINT;
 { - funkcja zmieniajaca czas podany jako tekst w formie hh:mm:ss na sekundy -
}

FUNCTION YTIME ( YY : LONGINT ) : STRING;
 { - funkcja zmieniajaca czas podany w sekundach na tekst w formie hh:mm:ss -
}

IMPLEMENTATION

 USES PRINTER,CRT,DOS,DIALOG,SIM1PROCES,SIM2PROCES,MAPA;
```

```
  FUNCTION  FC TYP_PROCESU: TYP_NAZWY_PROCESU;
              NUMER_PROCESU: INTEGER): INTEGER;
    BEGIN

    ...

    END;

  FUNCTION XTIME ( XX : STRING ) : LONGINT;
    BEGIN
{funkcja zmieniajaca czas podany jako teks w formie hh:mm:ss na sekundy}
    END; { XTIME }

  FUNCTION YTIME( YY : LONGINT ) : STRING;
    BEGIN
{funkcja zmieniajaca czas podany w sekundach na tekst w formie hh:mm:ss}
    END; { YTIME }

  PROCEDURE PROCES;

  {-------------------------------------------------------------------}

   PROCEDURE PORZADKOWANIE;

   VAR
    I : INTEGER;

   BEGIN
   FOR I:=1 TO MAX_PROCES_WYGENEROWANY DO
     IF PROCESY[I].CZY_USUNAC THEN
       BEGIN
       PROCESY[I].P:=PROCESY[MAX_PROCES_WYGENEROWANY].P;
       PROCESY[I].TYP:=PROCESY[MAX_PROCES_WYGENEROWANY].TYP;
       PROCESY[I].PLANOWY_CZAS_AKTYW:=
          PROCESY[MAX_PROCES_WYGENEROWANY].PLANOWY_CZAS_AKTYW;
       PROCESY[I].CZY_USUNAC:=FALSE;
       PROCESY[I].NR_OBJ_DANEGO_TYPU:=
          PROCESY[MAX_PROCES_WYGENEROWANY].NR_OBJ_DANEGO_TYPU;
       DEC(MAX_PROCES_WYGENEROWANY);
       END;
    PORZADKOWANIE_TABLICY_PROCESOW:=FALSE;
    END; { PROCEDURA PORZADKOWANIE }

   PROCEDURE WYBOR_PROCESU ;

   VAR
      I,MINIMALNY_CZAS: INTEGER;

    BEGIN
    MINIMALNY_CZAS:=PROCESY[1].PLANOWY_CZAS_AKTYW;
    I:=1;
    IND:=1;
    WHILE I < MAX_PROCES_WYGENEROWANY DO
      BEGIN
      I:=I+1;
      IF PROCESY[I].PLANOWY_CZAS_AKTYW < MINIMALNY_CZAS THEN
        BEGIN
        MINIMALNY_CZAS:=PROCESY[I].PLANOWY_CZAS_AKTYW;
        IND:=I
        END;
      END;
    IF  MINIMALNY_CZAS >= 0
    THEN
      BEGIN
      CZAS_AKTUALNY := CZAS_AKTUALNY + MINIMALNY_CZAS;
      FOR I:=1 TO MAX_PROCES_WYGENEROWANY DO
        PROCESY[I].PLANOWY_CZAS_AKTYW:=
```

```
          PROCESY[I].PLANOWY_CZAS_AKTYW-MINIMALNY_CZAS;
      END
    ELSE
      WRITELN('              BLAD W WYBORZE PROCESU ');
    IF TRYB_PRACY_MAPY THEN BEGIN
                            TWRITE('');
                            WRITELN('WYBRANO PROCES ',IND);
                            END;
    IF TRYB_PRACY_MAPY THEN WYBOR_OKNA ELSE WYBOR_OKNA_MAPY;
    CASE PROCESY[IND].TYP OF
      POCIAG: PROCES1(PROCESY[IND].P);
      STEROWANIE: PROCES2(PROCESY[IND].P);
      ZWOLNIENIE_PRZEBIEGU: PROCES3(PROCESY[IND].P);
      ELSE
       BEGIN
       TWRITE('');
       WRITELN('NIEZIDENTYFIKOWANY PROCES')
       END;
    END; { CASE }
END; { WYBOR_PROCESU }
{-------------------------------------------------------------------}


BEGIN
  { generowanie procesow typu POCIAG }

  FOR I1:=1 TO MAX_L_POCIAGOW DO
   BEGIN
   NEW(P1);
   P1^.N:=I1;
   P1^.E:=1;
   P1^.NUMER_ZADANIA:=I1;
   P1^.AKTUALNY_NUMER:=0;
   P1^.WSKAZNIK_RJ:=1;
   P1^.STAN:=1;
   P1^.KOLEJNY_ODCINEK_PRZEBIEGU:=0;
   P1^.NR_PRZEB:=0;
   P1^.KOLEJNY_NR_POC_W_ZADANIU:=1;
   P1^.KKK:=0;
   P1^.SEMAFOR:=TRUE;
   P1^.NR_POC_W_KOLEJCE:=1;
   P1^.KIERUNEK_JAZDY:=0;
   P1^.NR_NAJBL_SYGNAL:=0;
   P1^.POPRZEDNI_SYGNALIZATOR:=0;
   I3:=1;
    FOR I2:=1 TO (dlugosc pliku (POCIAGI+NR_ROZKLADU) ) DO
     BEGIN
     IF ZADANIA[I1,1]=(POCIAGI+NR_ROZKLADU)[I2,1]
      THEN BEGIN
           P1^.ROZKLAD_JAZDY[I3]:=(POCIAGI+NR_ROZKLADU)[I2,2];
           P1^.CZAS_JAZDY:=(POCIAGI+NUMER_ROZKLADU)[I2,3];
           I3:=I3+1;
           END;
     END;
   P1^.POLOZENIE_CZOLA:=0;

   PROCESY[I1].TYP:=POCIAG;
   PROCESY[I1].PLANOWY_CZAS_AKTYW:=P1^.CZAS_JAZDY[1]-300;
   PROCESY[I1].P:=P1;
   PROCESY[I1].CZY_USUNAC:=FALSE;
   PROCESY[I1].NR_OBJ_DANEGO_TYPU:=I1;
   END;

   AKTUALNA_LICZBA_WYGENEROWANYCH_OBIEKTOW_TYPU_POCIAG:=MAX_L_POCIAGOW;
   AKTUALNA_LICZBA_WYGENEROWANYCH_OBIEKTOW_TYPU_STEROWANIE:=0;
```

```
    AKTUALNA_LICZBA_WYGENEROWANYCH_OBIEKTOW_TYPU_ZWOLNIENIE_PRZEBIEGU:=0;
    MAX_PROCES_WYGENEROWANY:=MAX_L_POCIAGOW;

    IF TRYB_OBSERWACJI=1 THEN TRYB_PRACY_MAPY:=FALSE;
    CASE TRYB_OBSERWACJI OF
     1:INITMAPA;
     2:OKNO_KOMUNIKATY;
     END; {CASE}
    { wybor procesu }
    WHILE CZAS_AKTUALNY < CZAS_ZAKONCZENIA_SYMULACJI DO
      BEGIN
      IF PORZADKOWANIE_TABLICY_PROCESOW THEN PORZADKOWANIE;
      IF  TRYB_CZASOWY=3 THEN
       REPEAT UNTIL KEYPRESSED;
      WYBOR_PROCESU;
      END;
    EXIT; { DO PROGRAMU GLOWNEGO }
  END;

BEGIN
WRITEC'PODAJ CZAS ZAKONCZENIA SYMULACJI ? --- ');
READLN(CZAS_ZAKONCZENIA_SYMULACJI);
WRITEC'PODAJ MAKSYMALNA ILOSC SKLADOW POCIAGOW NA LINII KRR ? --- ');
READLN(MAX_L_POCIAGOW);
END. { PROCES }


UNIT SIM1PROCES;
{ Modul PROCES1 }

INTERFACE

 TYPE

   REKORD_STANU_PROCESU1_=^REKORD_STANU_PROCESU1;
   REKORD_STANU_PROCESU1 = RECORD
                             E: BYTE; { wartosc etykiety kontynuacji }
                             N: INTEGER; { kolejny numer obiektu typu pociag}
                NUMER_ZADANIA: INTEGER;
              AKTUALNY_NUMER: INTEGER;
               ROZKLAD_JAZDY: ARRAY[1..100] OF INTEGER;
                  CZAS_JAZDY: ARRAY[1..100] OF INTEGER;
                WSKAZNIK_RJ: BYTE;
                       STAN: BYTE;  {1:stoi, 2:jedzie, 3:stoi w kolejce}
            NR_NAJBL_SYGNAL: INTEGER;
      POPRZEDNI_SYGNALIZATOR: INTEGER;
            POLOZENIE_CZOLA: INTEGER;      { wartosc dodatnia:torowy,
                                             ujemna:zwrotnicowy}
              KIERUNEK_JAZDY: BYTE;        { 1: PYSKOWICE   2: BOGUCICE }
   KOLEJNY_ODCINEK_PRZEBIEGU: BYTE;
                 NR_PRZEB: INTEGER;       {0: brak przebiegu}
    KOLEJNY_NR_POC_W_ZADANIU: BYTE;
                      KKK: INTEGER;    {poprzedni przebieg, jeszcze nie
                                        zwolniony}
                   SEMAFOR: BOOLEAN;
           NR_POC_W_KOLEJCE: BYTE;      {1,2,3, itd   }
                      END;

 VAR
   P1:       REKORD_STANU_PROCESU1_; { unikalna nazwa }
   AKTUALNA_LICZBA_WYGENEROWANYCH_OBIEKTOW_TYPU_POCIAG:INTEGER;

PROCEDURE PROCES1 (W: REKORD_STANU_PROCESU1_);

IMPLEMENTATION
```

```
USES PRINTER,DIALOG,SIM2PROCES,SIMULATION,MAPA,POMOC,BAZA_KRR,AUP;

  {-------------------------------------------------------------------}
   PROCEDURE PROCES1 (W: REKORD_STANU_PROCESU1_);
     CONST
          ET1_=1; ET2_=2; ET3_=3;
          CZAS_DOJAZDU_DO_IZOLACJI=4;
     LABEL ET1, ET2, ET3; { etykiety kontynuacji }
     VAR
         I  :INTEGER;
         KRZ:INTEGER;


     BEGIN
       WITH W^ DO
         BEGIN
           CASE E OF { wymog techniczny w PASCAL-u }
             ET1_: GOTO ET1;
             ET2_: GOTO ET2;
             ET3_: GOTO ET3;
             ELSE  BEGIN
                 {  TWRITEC'    ');                              }
                  { WRITELNC'    BLAD KONTYNUACJI W PROCESIE POCIAG');  }
                   HALT
                   END;
           END; { CASE }



  ET1: { zgloszenie pociagu do systemu 5 minut przed planowym odjazdem}
       { z toru postojowego lub zwrotnego                             }
    {.................aktualne parametry pociagu.......................}
         SZUKAJ_PRZEBIEGU(POLOZENIE_CZOLA,N);
       E := ET2_; EXIT;                                    { 5 minut ^ }
{-------------------------------------------------------------------------}


  ET2: { odjazd pociagu - miniecie kolejnego sygnalizatora }


  ET3: { pociag mija kolejna izolacje }
       { ostatnia os pociagu zwalnia przebiegi }
 END; { PROCES1 }
  {-------------------------------------------------------------------}


BEGIN

END. { SIM1PROCES }


UNIT SIM2PROCES;
{ Modul PROCES2 }

INTERFACE

 TYPE

   REKORD_STANU_PROCESU2_=^REKORD_STANU_PROCESU2;
   REKORD_STANU_PROCESU2 = RECORD
                             E : BYTE;
                             N : INTEGER;
                    NR_PRZEB: INTEGER;
                  WYSWIETL_SZ: BOOLEAN;
                         END;
```

```
VAR
   P2:       REKORD_STANU_PROCESU2_;  { w zasadzie powinna byc unikalna nazwa }
   AKTUALNA_LICZBA_WYGENEROWANYCH_OBIEKTOW_TYPU_STEROWANIE:INTEGER;

PROCEDURE PROCES2 (W: REKORD_STANU_PROCESU2_);

IMPLEMENTATION

USES PRINTER,DIALOG,SIMULATION,SIM1PROCES,MAPA,POMOC,BAZA_KRR;

  {-------------------------------------------------------------------------}
   PROCEDURE PROCES2 (W: REKORD_STANU_PROCESU2_);
     CONST ET1_=1;ET2_=2;ET3_=3;ET4_=4;ET5_=5;
           MAX_LICZBA_SYGNALOW_AUP=13;
           MAX_LICZBA_SPRZECZNYCH_PRZEBIEGOW=35;
           CZAS_OCZEKIWANIA=10;      { czas testowania }
           CZAS_REAKCJI_ZSP=1;

     LABEL ET1,ET2,ET3,ET4,ET5; { etykiety kontynuacji }
     VAR
                  ETY:INTEGER;
                  I  :INTEGER;
         OCZEKIWANIE:BOOLEAN;
         ZAJMOWANIE :BOOLEAN;
         SYGNALY    :ARRAY[1..MAX_LICZBA_SYGNALOW_AUP] OF STRING[10];
         SPRZECZNE  :ARRAY[1..MAX_LICZBA_SPRZECZNYCH_PRZEBIEGOW] OF INTEGER;

     BEGIN
       WITH W^ DO
         BEGIN
           CASE E OF { wymog techniczny w PASCAL-u }
             ET1_: GOTO ET1;
             ET2_: GOTO ET2;
             ET3_: GOTO ET3;
             ET4_: GOTO ET4;
             ET5_: GOTO ET5;
             ELSE  BEGIN
                     TWRITEC'');}

                   WRITELNC'BLAD KONTYNUACJI W PROCESIE STEROWANIE'); HALT
                   END;
           END; { CASE }

  ET1: { kontrola,czy mozna ustawic caly przebieg   }
  ET2:   {  zajmowanie odcinkow przebiegu }
  ET3:   {zajecie przebiegow sprzecznych do niego }
  ET4:  { utwierdzenie przebiegu   }
  ET5:       {  nastawienie sygnalu zezwalajacego}
 {.......... samounicestwienie procesu ..........................}
 END; { PROCES2 }
  {-------------------------------------------------------------------}
BEGIN

END  { MPROCES2 }


UNIT SIM3PROCES;
{ Modul PPROCES3 }

INTERFACE

 TYPE

   REKORD_STANU_PROCESU3_=^REKORD_STANU_PROCESU2;
   REKORD_STANU_PROCESU3 = RECORD
```

```
                              E : BYTE;
                              N : INTEGER;
                  NUMER_ODCINKA: INTEGER;
                NUMER_PRZEBIEGU: INTEGER;
            PRZEBIEG_SAMOCZYNNY: INTEGER;
                      NUMER_SEM: INTEGER;
                          END;

VAR
   P3:       REKORD_STANU_PROCESU3_;  { w zasadzie powinna byc unikalna nazwa }
   AKTUALNA_LICZBA_WYGENEROWANYCH_OBIEKTOW_TYPU_ZWOLNIENIE_PRZEBIEGU:INTEGER;

PROCEDURE PROCES3 (W: REKORD_STANU_PROCESU3_);

IMPLEMENTATION

USES PRINTER,DIALOG,SIM1PROCES,SIMULATION,POMOC,BAZA_KRR;

  {-------------------------------------------------------------------------}
   PROCEDURE PROCES3 (W: REKORD_STANU_PROCESU3_);
     CONST ET1_=1;
     LABEL ET1; { etykiety kontynuacji }
     VAR
        ETY,J,I           : INTEGER;
        SEM_1,SEM_2,SEM-3: INTEGER;
        OD_1,OD_2         : INTEGER;
     BEGIN
       WITH W^ DO
        BEGIN
          CASE E OF { wymog techniczny w PASCAL-u }
            ET1_: GOTO ET1;
            ELSE  BEGIN
                   TWRITEC'');}
                  WRITELNC'BLAD KONTYNUACJI W PROCESIE 2'); HALT END;
          END; { CASE }


  ET1:

     IF ((NUMER_ODCINKA<>0) AND (NUMER_PRZEBIEGU>0))
      THEN     { zwolnienie odcinka i przebiegu          }
  {.....................................................................}
     IF ((NUMER_ODCINKA <> 0) AND (NUMER_PRZEBIEGU = 0))
      THEN BEGIN  { zwolnienie samego odcinka }
                  { obsluga blokady SBL }
  ..........SAMOUNICESTWIENIE SIE PROCESU  ..........................}


   END; { PROCES3 }
  {-------------------------------------------------------------------------}


BEGIN

END. { MPROCES3 }


UNIT MAPA;

INTERFACE
USES
 CRT,GRAPH,MAPA1,MAPA2;

CONST
 TRYB_PRACY_MAPY:BOOLEAN=TRUE; { TRUE-MAPA STATYCZNA,FALSE-MAPA DYNAMICZNA }
```

```
PROCEDURE BAZA; { ladowanie bazy }
PROCEDURE INITMAPA;  { inicjacja grafiki oraz pokazywania mapy }
PROCEDURE CLOSEMAPA; { wyjscie z grafiki }
PROCEDURE AKTUAL(STANY:TAB); { aktualizacja stanow elementow mapy
                               definicja tablicy STANY oraz typu TAB
                               w module mapa2 }


IMPLEMENTATION


END. { MAPA }

UNIT MAPA1;

INTERFACE

  CONST
        MAX_LSEM=450;       { Max. liczba semaforow na mapie              }
        MAX_LODCT=270;      { Max. liczba ODCINKOW TOROWYCH               }
        MAX_LODCZ=300;      { MAX. LICZBA ODCINKOW ZWROTNICOWYCH          }
        MAX_LPER=35;        { MAX. LICZBA PERONOW                         }
        MAX_LKOR=70;        { MAX. LICZBA POL KOREKT                      }
        MAX_LSTER=80;       { MAX. LICZBA POL STERUJACYCH                 }
        MAX_LWLOT=20;       { MAX. LICZBA WLOTOW                          }
        MAX_LPODZ=500;      { MAX. LICZBA ODCINKOW PODZIALEK              }
        MAX_NADZ=250;       { MAX. LICZBA WIERSZY STEROWANIA ODC ZWROTNICOWYMI

        MAX_STACJI=33;
        LK_MAPA_ODCT=6;   { Liczba kolumn tablicy MAPA_ODCT             }
        LK_MAPA_ODCZ=6;   { Liczba kolumn tablicy MAPA_ODCZ             }
        LK_MAPA_SEM=5;    { Liczba kolumn tablicy MAPA_SEM              }
        LK_MAPA_PER=4;    { Liczba kolumn tablicy MAPA_PER              }
        LK_MAPA_KOR=5;    { Liczba kolumn tablicy MAPA_KOR              }
        LK_MAPA_STER=5;   { Liczba kolumn tablicy MAPA_STER             }
        LK_MAPA_WLOT=4;   { Liczba kolumn tablicy MAPA_WLOT             }
        LK_MAPA_PODZ=3;   { Liczba kolumn tablicy MAPA_PODZ             }
        LK_MAPA_NADZ=15;  { Liczba kolumn tablicy MAPA_NADZ             }
        PX=1.15;                { Wspolczynnik korekcji ekranu po osi X }
        PY=0.75;                { Wspolczynnik korekcji ekranu po osi Y }
        LIT_TXT=2;              { PIERWOTNE SKALE DLA TEKSTU,OPISU SEMAFOROW }
        LIT_SEM=0.9;            { ORAZ ODCINKOW TOROWYCH }
        LIT_ODCT=1;

  TYPE
        MAPA_ODCT_  = ARRAY[1..MAX_LODCT, 1..LK_MAPA_ODCT]  OF INTEGER;
        MAPA_ODCZ_  = ARRAY[1..MAX_LODCZ, 1..LK_MAPA_ODCZ]  OF INTEGER;
        MAPA_SEM_   = ARRAY[1..MAX_LSEM,  1..LK_MAPA_SEM]   OF INTEGER;
        MAPA_KOR_   = ARRAY[1..MAX_LKOR,  1..LK_MAPA_KOR]   OF INTEGER;
        MAPA_STER_  = ARRAY[1..MAX_LSTER, 1..LK_MAPA_STER]  OF INTEGER;
        MAPA_WLOT_  = ARRAY[1..MAX_LWLOT, 1..LK_MAPA_WLOT]  OF INTEGER;
        MAPA_PER_   = ARRAY[1..MAX_LPER,  1..LK_MAPA_PER]   OF INTEGER;
        MAPA_PODZ_  = ARRAY[1..MAX_LPODZ, 1..LK_MAPA_PODZ]  OF INTEGER;
        MAPA_NADZ_  = ARRAY[1..MAX_NADZ, 1..LK_MAPA_NADZ]  OF INTEGER;
        MENU_TXT    = STRING[10];
        NAZWA_ST    = STRING[24];
        MAPA_ST_    = ARRAY[1..MAX_STACJI] OF NAZWA_ST;

  VAR
        MAPA_ODCT  : ^MAPA_ODCT_;  { Mapa  statyczna odcinkow torowych     }
        MAPA_ODCZ  : ^MAPA_ODCZ_;  { Mapa  statyczna odcinkow zwrotnicowych }
        MAPA_SEM   : ^MAPA_SEM_;   { Mapa semaforow
        MAPA_KOR   : ^MAPA_KOR_;   { MAPA POL KOREKT }
```

```
      MAPA_STER  : ^MAPA_STER_; { MAPA POL STERUJACYCH
      MAPA_WLOT  : ^MAPA_WLOT_; { MAPA WLOTOW }
      MAPA_PER   : ^MAPA_PER_;  { MAPA PERONOW }
      MAPA_PODZ  : ^MAPA_PODZ_; { MAPA PODZIALEK }
      MAPA_ST    : ^MAPA_ST_;   { MAPA STACJI  TEKST}
      MAPA_NADZ  : ^MAPA_NADZ_; { MAPA NADZORU ODC ZWROTNICOWYCH }
      LSEM       : INTEGER;     { Liczba semaforow na mapie          }
      LODCT      : INTEGER;     { Liczba odcinkow torowych }
      LODCZ      : INTEGER;     { Liczba odcinkow zwrotnicowych }
      LPER       : INTEGER;     { Liczba peronow }
      LKOR       : INTEGER;     { Liczba pol korekt }
      LSTER      : INTEGER;     { Liczba pol sterujacych }
      LWLOT      : INTEGER;     { Liczba wlotow }
      LPODZ      : INTEGER;     { Liczba odc podzialek }
      LNADZ      : INTEGER;     { Liczba rekordow sterujacych przebiegami }
      NRSEM      : INTEGER;     { Numer semafora                     }
      NR         : INTEGER;     { Numer zmienna uniwersalna }
      NRODCT     : INTEGER;     { Numer odc torowego }
      NRODCZ     : INTEGER;     { Numer odc zwrotnicowego }
      KIERUNEK   : INTEGER;     { Kierunek rysowania semafora 0-lewo 1-prawo

      DLPER      : INTEGER;     { Dlugosc peronu }
      SZERPER    : INTEGER;     { Szerokosc peronu }
      STAN       : INTEGER;     { Atrybut stanu elementu             }
      XMIN,YMIN  : LONGINT;     { Wspolrzedne poczatku ekranu na mapie }
      XMAX,YMAX  : LONGINT;     { Wspolrzedne konca    ekranu na mapie }
      SZER_MAP   : LONGINT;     { Bazowa  szerokosc  mapy (po osi X)   }
      WYS_MAP    : LONGINT;     { Bazowa  wysokosc  mapy (po osi Y)    }
      I,J        : INTEGER;     { nmienne uniwersalne }
      STR_.S     : INTEGER;     { Biezacy numer strony karty Hercules  }
      SKALA      : REAL;        { Ogolna skala mapy                    }
      SKALA_X    : REAL;        { Biezaca skala mapy (powiekszenie X )  }
      SKALA_Y    : REAL;        { Biezaca skala mapy (powiekszenie Y )  }
      INKX,INKY  : LONGINT;     { Inkrement przesuwu mapy klawiszami   }
      NINKX,NINKY:LONGINT;      { Jak INKX i INKY tylko bez korekcji ek. }
      XM,YM      : LONGINT;     { Wspolrzedne srodka ekranu na mapie   }
      NXM,NYM    : LONGINT;     { Jak XM i YM tylko bez korekcji ekranu }
      STAC       : NAZWA_ST;    { Nazwa stacji }
      TEKST      : STRING[80];  { pomocnicza }
      TEKST1     : STRING[80];  { pomocnicza }
      PLIK_MAPA  : STRING[80];  { nazwa pliku-bazy danych graficznych mapy }
      STEROWNIK  : INTEGER;
      TRYB       : INTEGER;
      ERRCODE    : INTEGER;
      F          : TEXT;        { plik zawierajacy mape-baze danych }
      X1,Y1,X2,Y2:INTEGER;      { Wspolrzedne uniwersalne }
      SZER_EKR   : INTEGER;
      WYS_EKR    : INTEGER;
      ESCMAP     : POINTER;     { Zmienna zawierajaca obraz wyciety przez
                                  ramke menu }
      TAB_MENU   : ARRAY [1..6] OF MENU_TXT; { Tablica zawierajaca tekstowe
                                               opcje menu }
    WSK_WYJ_MAPY: BOOLEAN;      {jezeli TRUE , dynamiczne wyjscie z mapy }

PROCEDURE GRAFIKA; { Inicjacja grafiki i przejscie do trybu tekstowego }
FUNCTION INKEY:INTEGER; { Czytanie kodu klawisza }
FUNCTION READG (XP,YP,N: INTEGER) : REAL; { Czytanie liczby w trybie
                                            graficznym}
PROCEDURE CHOWAJ_RAMKE;
PROCEDURE PODSTAW;
PROCEDURE PODSWIETLC(W,K:INTEGER);
PROCEDURE ZGASC(W,K:INTEGER);
PROCEDURE PODSWIETLSC(W,K,L:INTEGER);
PROCEDURE ZGASSC(W,K,L:INTEGER);
PROCEDURE STACJA; { procedure wyboru stacji do podgladu }
PROCEDURE OKNO_ESC;
```

```
 PROCEDURE ESC_MENU;
 PROCEDURE SKALA_LITER(SKALA:REAL);
 PROCEDURE EKRAN;
 FUNCTION STRONA: INTEGER;
 PROCEDURE RYSUJ (X1,Y1,X2,Y2,NR,STAN: INTEGER);
 PROCEDURE RYSUJ_ZW (X1,Y1,X2,Y2,NR,STAN: INTEGER);
 PROCEDURE RYSUJ_STER (X1,Y1,X2,Y2: INTEGER);
 PROCEDURE RYSUJ_KOR (X1,Y1,X2,Y2: INTEGER);
 PROCEDURE RYSUJ_PERON(X1,Y1,DL,SZER,M:INTEGER;NS:STRING);
 PROCEDURE RYSUJ_WLOT(X1,Y1,KIER:INTEGER);
 PROCEDURE RYSUJ_SEMAFOR(X1,Y1,NR,KIER,STAN:INTEGER);
 PROCEDURE RYSUJ_PODZ(X1,X2,NR: INTEGER);
 PROCEDURE RAMKA; { ramka ekranu }
 PROCEDURE NAPIS; { napis na gorze ramki ekranu F5 ... }

IMPLEMENTATION
USES
 GRAPH,CRT;


END.{MAPA1}


UNIT MAPA2;

INTERFACE
TYPE
 TAB = ARRAY[1..4] OF INTEGER;

VAR
 STANY: TAB;

PROCEDURE LAD_MAPA; { ladowanie bazy danych }
PROCEDURE SKALAM;   { zmiana skali mapy }
PROCEDURE KROKM;    { -//-   kroku przesuwu }
PROCEDURE WIDOK;    { widok xy }
PROCEDURE MENU;
PROCEDURE POK_EKRAN; { pokazanie ekranu }
PROCEDURE MENU_STRZALKI;  { obsluga strzalek }
PROCEDURE PRZESKALOWANIE; { przeskalowanie bazy danych }
PROCEDURE WSTEP;{ nadanie wartosci poczatkowych dla zmiennych graficznych
                  mapy}
PROCEDURE PETLA;{ petla oczekiwania na klawisz }

IMPLEMENTATION

USES
 CRT,GRAPH,MAPA1;

END. { MAPA2 }


UNIT BAZA_KRR;
   { baza danych dla modelu }

   INTERFACE

   CONST

   DLUGOSC_POCIAGU = 200;
   MAX_DL_RJ = 120;

   LICZBA_SKLADOW_NA_KRR = 40;
   LICZBA_NUMEROW_SKLADU = 15;

   L_NAZ_G = 103;
```

```
L_SYGN  = 449;
L_TOR_OD_IZ = 259;
L_P_STER = 84;
L_P_KOR = 62;
L_L_STER = 41;
L_PRZEB = 234;
L_ZWR = 58;
L_ZWR_OD_IZOL = 87;
L_WJAZDOW = 15;
MAX_L_PRZEB_SPRZECZ = 25;
LKOL1=2;
LKOL2=7;
LKOL3=3;
LKOL4=16;
LKOL5=4;
LKOL6=8;
LKOL7=5;

TYPE
    NAZWY_GEOGRAFICZNE=ARRAY [1..L_NAZ_G,1..LKOL1] OF STRING[35];
    RODZAJ_SYGNALIZATOROW=ARRAY[1..L_SYGN,1..LKOL2] OF INTEGER;
    OPIS_SYGNALIZATOROW=ARRAY[1..L_SYGN,1..LKOL3] OF STRING[6];
    ODCINKI_TOROWE=ARRAY[1..L_TOR_OD_IZ,1..LKOL1] OF STRING[5];
    ODCINKI_TOROWE_OPIS=ARRAY[1..L_TOR_OD_IZ,1..LKOL4] OF INTEGER;
    POLA_KOREKT=ARRAY[1..L_P_KOR] OF BYTE;
    POLA_STERUJACE=ARRAY[1..L_P_STER,1..LKOL5] OF INTEGER;
    POLA_LOKALNE=ARRAY[1..L_L_STER,1..LKOL3] OF INTEGER;
    PRZEBIEGI=ARRAY[1..L_PRZEB,1..LKOL6] OF INTEGER;
    ODCINKI_IZOLOWANE_PRZEBIEGU=ARRAY[1..L_PRZEB,1..LKOL6] OF INTEGER;
    ZWROTNICE_PRZEBIEGU=ARRAY[1..L_PRZEB,1..LKOL7] OF INTEGER;
    STAN_ZWROTNIC_PRZEBIEGU=ARRAY[1..L_PRZEB,1..LKOL7] OF INTEGER;
    STAN_IZOLACJI_ZWROTNICOWYCH=ARRAY[1..L_ZWR_OD_IZOL] OF INTEGER;
    WEJSCIA_KRR=ARRAY[1..L_WJAZDOW] OF INTEGER ;
    ZWROTNICE=ARRAY[1..L_ZWR,1..LKOL3] OF  STRING[10];
    IZOLACJA_ZWROTNIC= ARRAY[1..L_ZWR_OD_IZOL,1..LKOL1] OF STRING[10];
    PRZEBIEGI_SPRZECZNE=ARRAY[1..L_PRZEB,1..MAX_L_PRZEB_SPRZECZ] OF
                                                        INTEGER;

VAR
   NAZ_GEOG:NAZWY_GEOGRAFICZNE;          {nazwy geograficzne linii krr}
   R_SYGNAL:RODZAJ_SYGNALIZATOROW;       {opis sygnalizatorow na linii
                                          krr}
   SYGNAL:OPIS_SYGNALIZATOROW;           {opis sygnalizatorow wg bpk}
   ODC_TOR:ODCINKI_TOROWE;               {opis odcinkow torowych wg bpk}
   TOR_IZOL:ODCINKI_TOROWE_OPIS;         {opis odcinkow torowych}
   POLA_KOR:POLA_KOREKT;                 {opis pol korekt na linii krr}
   POLA_STER:POLA_STERUJACE;             {opis pol sterujacych na linii
                                          krr}
   POLA_LOK:POLA_LOKALNE;                {opis lokalnych pol sterujacych,
                                          tylko dla potrzeb modelu}
   PRZEB:PRZEBIEGI;                      {opis przebiegow na linii krr}
   ODC_IZOL:ODCINKI_IZOLOWANE_PRZEBIEGU; {opis wszyskich odcinkow
                                          izolowanych przebiegu}
   ZWROT_PRZEBIEGU:ZWROTNICE_PRZEBIEGU ; {opis kolejnych zwrotnic
                                          przebiegu}
   STAN_ZWR_PRZEBIEGU:STAN_ZWROTNIC_PRZEBIEGU;
   STAN_IZOL_ZWROT:STAN_IZOLACJI_ZWROTNICOWYCH;
   WESIEC:WEJSCIA_KRR;
   ZWROT:ZWROTNICE;
   IZOL_ZWROT:IZOLACJA_ZWROTNIC;
   PRZEB_SPRZECZNE:PRZEBIEGI_SPRZECZNE;
   SEM_TYPU_S:ARRAY [1..16] OF STRING[1];
   SEM_TYPU_L:ARRAY [1..6] OF STRING[1];
   SEM_TYPU_LW:  ARRAY [1..6] OF STRING[1];
   SEM_TYPU_STM:  ARRAY [1..4] OF STRING[1];
   MAX_L_POCIAGOW:INTEGER;
```

```
      MAX_L_NUMEROW: INTEGER;
      ZADANIA: ARRAY[1..LICZBA_SKLADOW_NA_KRR,1..LICZBA_NUMEROW_SKLADU] OF
                                                                  INTEGER;
      NR_ROZKLADU: BYTE;

   IMPLEMENTATION
   USES CRT,DOS;


   END. {BAZA_KRR}

UNIT AUP;

INTERFACE


PROCEDURE SZUKAJ_PRZEBIEGU(NRPOLSTER,NUMER_P);
PROCEDURE WYSWIETL_SYGNAL(KIER_JAZDY,NUMER_POLA_LOKALNEGO);



IMPLEMENTATION

USES SIM1PROCES,SIM2PROCES,BAZA_KRR;

 BEGIN
 ...
 END.   {AUP}

UNIT POMOC;

{ ------------------------------------------------------------------------
  Modul zawierajacy procedure umozliwiajaca wymiane danych pomiedzy symula-
  torem ruchu pociagow a systemem SNB/AUP , a rowniez procedure zerujaca
  rekord transportowy przechowujacy odczytane badz zapisywane dane.
  ------------------------------------------------------------------------ }

INTERFACE

 USES CRT;

 CONST
   MAX_LICZ_PL_NUMER = 5;              { maksymalna liczba pol numerycznych }
   MAX_LICZ_PL_ZNAK  = 9;              { maksymalna liczba pol znakowych    }
   MAX_LICZ_PL_LOGIC = 2;              { maksymalna liczba pol logicznych   }

 TYPE
   TYP_REK = RECORD                    { typ rekordu transportowego
               PL_NUMER : ARRAY[1..MAX_LICZ_PL_NUMER] OF INTEGER;
               PL_ZNAK  : ARRAY[1..MAX_LICZ_PL_ZNAK]  OF STRING[210];
               PL_LOGIC : ARRAY[1..MAX_LICZ_PL_LOGIC] OF BOOLEAN;
               PL_DATA  : STRING[8];
             END;

 VAR
   REK_TRANSPORTOWY : TYP_REK;         { rekord transportowy sluzy do przeka-
                                         zywania danych pomiedzy proceduram
                                         odczytujacymi/zapisujacymi dane z/do
                                         zbioru dBase zawierajacego dane

 PROCEDURE ZEROWANIE;
 PROCEDURE KONWERSJA(NAZWA  : STRING;  { nazwa zbioru z baza danych
                     NR_REK : INTEGER; { numer rekordo w bazie danych
                     ZNACZN : BYTE);   { znacznik operacji na zbiorze
```

```
                                    1 - odczyt ze zbioru
                                    2 - zapis do zbioru
                                    3 - dopisanie na koncu zbioru      >

IMPLEMENTATION

  ------------------------------------------------------------------------ >


  BEGIN
  END. < POMOCNICZY >
```

4. Uwagi końcowe

Zrealizowana dla potrzeb modelu symulacyjnego struktura symulacyjna charakteryzuje się prostą realizacją oraz niewyszukanymi metodami informatycznymi z punktu widzenia kosztów algorytmów, ponieważ głównym celem było zrealizowanie samego zintegrowanego modelu ruchu pociągów. Aktualnie moduł SIMULATION nie jest jeszcze modułem uniwersalnym, ponieważ konieczna jest ingerencja w jego treść dla każdego realizowanego programu symulacyjnego. Można jednak stwierdzić, że dla celów realizacji modelu ruchu pociągów jest to już wersja wystarczająca.

Realizacja przedstawionej w artykule struktury oprócz możliwości symulacyjnych udostępnia pełne możliwości graficzne języka Turbo PASCAL.

Realizacja uniwersalnego modułu SIMULATION oraz pełnego zestawu procedur pomocniczych ułatwiających programowanie symulacyjne będzie realizowane w przyszłości, być może nawet w postaci pakietu zintegrowanego z symulacyjnym edytorem strukturalnym. Na tym etapie prac szczególnie istotna będzie minimalizacja kosztów realizacji procedur sterujących symulacją.

LITERATURA

[1] KRAWIEC S: Prezentacja ogólna modelu ruchu pociągów. Zeszyty Naukowe Politechniki Śląskiej, s.Transport nr 13, Gliwice 1989.
[2] KRAWIEC S: Opis nieformalny modelu ruchu pociągów - elementy. Zeszyty Naukowe Politechniki Śląskiej,s.Transport nr 13,Gliwice 1989.
[3] KRAWIEC S: Opis nieformalny modelu ruchu pociągów - interakcja elementów. Zeszyty Naukowe Politechniki Śląskiej, s.Transport nr 13, Gliwice 1989.
[4] KRAWIEC S: Opis formalny modelu ruchu pociągów. Zeszyty Naukowe Politechniki Śląskiej, s.Transport nr 13, Gliwice 1989.
[5] JANECKI R.: Baza danych topologicznych i ruchowych dla potrzeb modelu symulacyjnego ruchu pociągów. Zeszyty Naukowe Politechniki Śląskiej, s.Transport nr 13, Gliwice 1989.

[6] KONIECZNY R. + zespół (praca zbiorowa): Zastosowanie języka LOGLAN do modelowania dużych systemów transportowych na przykładzie modelu ruchu pociągów - Katowice 1987 (maszynopis pracy naukowo-badawczej NB-277/RT/87 program RP.I.09)
[7] KONIECZNY R. + zespół (praca zbiorowa): Zastosowanie języka LOGLAN do modelowania dużych systemów transportowych na przykładzie modelu ruchu pociągów część II - Katowice 1988 (maszynopis pracy naukowo-badawczej NB-195/RT/88 program RP.I.09)
[8] KONIECZNY R. + zespół (praca zbiorowa): Zastosowanie języka LOGLAN do modelowania dużych systemów transportowych na przykładzie modelu ruchu pociągów część III - Katowice 1989 (maszynopis pracy naukowo-badawczej NB-195/RT/88 program RP.I.09).
[9] KONIECZNY R. + zespół (praca zbiorowa): Moduły programowe wspomagające komputerowy makromodel ruchu pociągów (maszynopis pracy naukowo-badawczej).
[10] BIELECKI J: Turbo PASCAL 5.0 - wersja profesjonalna - Wydawnictwa Komunikacji i Łączności, Warszawa 1989.
[11] CIRIC BOBBY, THIES KLAUS DIETER: Turbo PASCAL 5.0/5.5 - te-wi Verlag GmbH 1989.
[12] KONIECZNY R.: Przykłady rozwiązania problemów symulacyjnych w języku LOGLAN. Zeszyt Naukowy Politechniki Śląskiej, s. Transport nr 13, Gliwice 1989.
[13] TISCHER: Turbo PASCAL intern. Data Becker 1989.
[14] ZEIGLER B.P.: Teoria modelowania i symulacji. PWN, Warszawa 1984.
[15] KONIECZNY R.,KRAWIEC S.: Zagadnienia komputerowej realizacji mapy rejonu sieci kolejowej - część I.(niniejszy zeszyt).
[16] KONIECZNY R.,KRAWIEC S.: Zagadnienia komputerowej realizacji mapy rejonu sieci kolejowej - część II.(niniejszy zeszyt).
[17] KRAWIEC S.,KONIECZNY R.,JANECKI R.: Dwukomputerowa realizacja makromodelu ruchu pociągów.(niniejszy zeszyt).
[18] KONIECZNY R.: Niektóre aspekty rozwoju modułu MAPA dla potrzeb symulatora ruchu pociągów.(niniejszy zeszyt).
[19] KONIECZNY R.: Zagadnienie realizacji loglanowskiego modułu SIMULATION na bazie języka Turbo PASCAL. (niniejszy zeszyt).

SIMULATING STRUCTURES REALIZATION IN TURBO PASCAL FOR SIMULATING MODEL OF RAILWAY TRAFFIC

Summary

Simulating structures realisation in v.5.0 Turbo PASCAL has been presented in the paper. Principles of SIMULATION module, SIMiPROCESS module, i=1,2,..,LPROCESS for a m structure and inter-object communication during simulation process realization have been discussed. The realized simulation structure as a simulated model of the structure has been used as an example for railway traffic simulation in Regional Railway Traffic System.

REALISIERUNG DER SIMULATIONSSTRUKTUREN IN DER PROGRAMMIERSPRACHE TURBO PASCAL FÜR DAS SIMULATIONSMODELL DES ZUGVERKEHRS

Zusammenfassung

Im Aufsatz wurde die Realisierung der Simulationsstruktur in der Programmiersprache TURBO PASCAL v.5.0 vorgestellt. Es wurden die Aufbauprinzipien der Programmodule SIMULATION und SIMiPROCES, bei i=1,...,L Prozeß für die oben genannte Struktur sowie die Kommunikationsprinzipien zwischen den Obiekten, die während der Realisierung des Simulationsprozesses gebildet werden, vorgestellt. Als Beispiel wurde die Struktur des Simulationsmodells für die Simulation des Zugverkehrs im Regionalen Bahnverkehr KRR.

СТРУКТУРА МОДЕЛИ ДВИЖЕНИЯ ПОЕЗДОВ НА ЯЗЫКЕ TURBO PASCAL

Резюме

В статьи представлено способ реализации структуры модели на языке Turbo PASCAL v.5.0. Представлено способы строения модуля SIMULATION, модуля SIMiPROCES, i=1,2,...,L процессов для выше упомянутой структуры а также принципы сообщении между получаемыми во время моделирования обектами. В характере примера уже практически использованной структуры модели подано модель движения поездов по линиям районной железной дороги.