ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1989
Seria: TRANSPORT z. 13 1034

Roman KONIECZNY

PRZYKŁADY ROZWIĄZANIA PROBLEMÓW SYMULACYJNYCH W JĘZYKU LOGLAN

Streszczenie. Artykuł niniejszy stanowi kontynuację opracowania [1] dotyczącego zasobów symulacyjnych języka LOGLAN. Przedstawione przykłady programów stanowić mogą pewien schemat budowy symulatorów systemów rzeczywistych. Przykładami symulacji rozważanymi w artykule są: obsługa abstrakcyjnego urządzenia, system o zasobach dzielonych oraz przejazd grupy pojazdów przez obszar kontrolowany.

Na podstawie zaprezentowanych w artykule przykładów programów, można stwierdzić dużą przydatność LOGLANu do budowy programów symulacyjnych dla różnego rodzaju zastosowań. Należy podkreślić dodatkowy fakt, że loglanowska klasa SIMULATION może być rozbudowana o nowe procedury (dalsze generatory liczb pseudolosowych, histogramy, wykresy wyników, itd.), a także o elementy symulacji mieszanej (dyskretno-ciągłej) lub wielokomputerowej. Badania nad tymi zagadnieniami są prowadzone aktualnie w Instytucie Transportu Politechniki Śląskiej.

1. Uwagi wstępne

Artykuł niniejszy stanowi kontynuację opracowania [1] dotyczącego zasobów symulacyjnych języka LOGLAN. Przedstawione przykłady programów stanowić mogą pewien schemat budowy symulatorów systemów rzeczywistych. Budowa symulatora składa się z dwóch podstawowych etapów:

- opisu symulowanego obiektu (systemu),
- implementacji, polegającej na odwzorowaniu opisu obiektu na zapis w postaci programu skonstruowanego w wybranym języku programowania.

Przykładami symulacji rozważanymi w artykule są: obsługa abstrakcyjnego urządzenia, system o zasobach dzielonych oraz przejazd grupy pojazdów przez obszar kontrolowany.

2. Opis symulowanego obiektu

Opis symulowanego obiektu (systemu) może być dokonany w sposób sformalizowany lub nieformalny. Zagadnieniom tym poświęcone są m.in. praca [7] oraz artykuły [2], [3], [4]. Uproszczony nieformalny opis obiektu rzeczywistego, który ma być odwzorowany w postaci symulatora, może być przedstawiony, jako następująca czwórka uporządkowana:

opis_nieformalny ≙ < LE, LZO, IE, SO >

gdzie:

LE - lista elementów,

LZO - lista zmiennych opisowych,

IE - interakcja elementów,

SO - schemat oddziaływań.

Lista elementów jest specyfikacją obiektów uwzględnionych w procesie symulacji systemu rzeczywistego. *Lista zmiennych opisowych* wyszczególnia atrybuty, zmienne stanu i parametry związane z danym obiektem (elementem). *Interakcja elementów* określa wzajemny wpływ, oddziaływanie lub działanie elementów modelu systemu. *Schemat oddziaływań* określa sposób powiązania elementów, nadrzędność - podległość itd.

3. Symulacja obsługi urządzenia

Rozważmy model symulacyjny składający się z kolejki klientów czekających na obsługę przez dane (abstrakcyjne) urządzenie. Interesuje nas obserwacja urządzenia przez pewien czas. Osoby zgłaszają się w losowych odstępach czasu i ustawiają się w kolejce do urządzenia.

LISTA ELEMENTÓW: *urządzenie* , *klient* ;

LISTA ZMIENNYCH OPISOWYCH:

urządzenie :

- czas_obsługi: REAL (czas obsługi klienta przez urządzenie);
- odpoczynek : REAL (łączny czas bezczynności urządzenia);
- przerwa : REAL (początek przerwy w pracy);
- obsługiwany : KLIENT (wskazuje obsługiwanego klienta);
- kolejka : FIFO (kolejka związana z urządzeniem);

klient :

- numer : INTEGER (numer bieżący klienta);
- czas_przyjścia: REAL (czas przybycia klienta);
- czas_obsługi : REAL (czas obsługi klienta);

INTERAKCJA ELEMENTÓW:

jeśli urządzenie jest bezczynne, to klient je uaktywnia .

Poniżej podano listing programu URZADZENIE realizującego rozważany problem:

```
PROGRAM URZADZENIE; (* Symulacja obslugi urzadzenia *)

(*$L-*)
(*.....tutaj jest tresc klasy SIMULATION *)
(*$L+*)

BEGIN
  PREF SIMULATION BLOCK;
```

```
UNIT URZADZENIE: SIMPROCESS CLASS(KOLEJKA: FIFO);
  VAR CZAS_OBSLUGI: REAL,
      ODPOCZYNEK  : REAL,
      PRZERWA     : REAL,
      OBSLUGIWANY : KLIENT;
   BEGIN
     PRZERWA:=TIME;
     DO
      ODPOCZYNEK:=ODPOCZYNEK-PRZERWA+TIME;
      WHILE NOT KOLEJKA.EMPTY
       DO
         OBSLUGIWANY:=KOLEJKA.FIRST;
         CZAS_OBSLUGI:=RANDOM*4,
         (* ... obsluga pierwszego klienta z kolejki ... *)
         CALL HOLD(CZAS_OBSLUGI);
         (* ... koniec obslugi klienta ... *)
         CALL KOLEJKA.OUTFIRST,
         (* ... uaktywnienie klienta ... *)
         CALL SCHEDULE(OBSLUGIWANY,TIME);
       OD;
      PRZERWA:=TIME;
      CALL PASSIVATE;
      (* ... URZADZENIE staje sie procesem zawieszonym ... *)
      (* ..... i czeka na uaktywnienie ... *)
   OD;
  END URZADZENIE;

UNIT KLIENT: SIMPROCESS CLASS(NR: INTEGER, U: URZADZENIE);
  VAR CZAS_PRZYJSCIA: REAL, (* Czas przyjscia klienta *)
      CZAS_OBSLUGI  : REAL; (* Czas obslugi   klienta *)
  BEGIN
    (* ... Przyjscie klienta ... *)
    CZAS_PRZYJSCIA:=TIME;
    WRITELN(F,TIME:5:2," Klient nr ",NR:3," wygenerowany");
    (* ... Klient ustawia sie na koncu kolejki ... *)
    CALL INTO(U.KOLEJKA);
    (* ... Jesli urzadzenie jest bezczynne... *)
    (* ... to klient je uaktywnia ... *)
    IF U.KOLEJKA.CARDINAL=1 THEN CALL SCHEDULE(U,TIME) FI;
    CALL PASSIVATE; (* ... Klient stoi w kolejce ... *)
    (* ... Teraz klient jest juz obsluzony ... *)
    CZAS_OBSLUGI:=TIME-CZAS_PRZYJSCIA;
    WRITELN(F,TIME:5:2," Klient nr ",NR:3,
      " byl obslugiwany ",CZAS_OBSLUGI:5:2);
  END KLIENT;

VAR (* Zmienne globalne *)
  U: URZADZENIE, CZAS_SYMULACJI: REAL, I: INTEGER,
  F: FILE;

  BEGIN (* Program glowny *)
    (* ... Otwarcie pliku dla rejestracji wynikow ... *)
    OPEN(F,TEXT,UNPACK("WYNIKI.DAT"));
    CALL REWRITE(F);
    (* ... Generowanie procesu URZADZENIE  . *)
    U:=NEW URZADZENIE(NEW FIFO);
    WRITELN(F,TIME:5:2,"  URZADZENIE WYGENEROWANE");
    I:=1; CZAS_SYMULACJI:=120 (* minut *);
    WHILE TIME<=CZAS_SYMULACJI
     DO
      (* ... Generowanie nowego klienta ... *)
      CALL SCHEDULE(NEW KLIENT(I,U), TIME); I:=I+1;
      (* ... Klienci przychodza w losowych odstepach czasu *)
      CALL HOLD(RANDOM*10);
     OD;
    (* ... Zakonczenie symulacji ... *)
```

```
    WRITELN(F,
     "Laczny czas bezczynnosci urzadzenia = ",U.ODPOCZYNEK:5:2);
  END;

END URZADZENIE.
```

Przebieg wykonania programu dla czasu symulacji równego 120 jednostek czasowych (np. minut) jest następujący:

```
IIUW LOGLAN-82  Concurrent Executor  Version 4.35
May 21, 1988
(C)Copyright  Institute of Informatics, University of Warsaw

  0.00   URZADZENIE WYGENEROWANE
  0.00  Klient nr   1  wygenerowany
  1.97  Klient nr   1  byl obslugiwany  1.97
  3.04  Klient nr   2  wygenerowany
  3.51  Klient nr   3  wygenerowany
  4.40  Klient nr   4  wygenerowany
  5.90  Klient nr   2  byl obslugiwany  2.86
  8.38  Klient nr   3  byl obslugiwany  4.87
 10.03  Klient nr   5  wygenerowany
 10.03  Klient nr   4  byl obslugiwany  5.63
 12.97  Klient nr   5  byl obslugiwany  2.95
 18.78  Klient nr   6  wygenerowany
 19.47  Klient nr   6  byl obslugiwany  0.69
 21.39  Klient nr   7  wygenerowany
 22.27  Klient nr   7  byl obslugiwany  0.88
 30.24  Klient nr   8  wygenerowany
 32.21  Klient nr   8  byl obslugiwany  1.98
 34.63  Klient nr   9  wygenerowany
 35.78  Klient nr   9  byl obslugiwany  1.14
 40.85  Klient nr  10  wygenerowany
 41.13  Klient nr  10  byl obslugiwany  0.28
 50.34  Klient nr  11  wygenerowany
 52.02  Klient nr  11  byl obslugiwany  1.68
 52.51  Klient nr  12  wygenerowany
 53.31  Klient nr  12  byl obslugiwany  0.80
 53.68  Klient nr  13  wygenerowany
 55.73  Klient nr  13  byl obslugiwany  2.06
 56.73  Klient nr  14  wygenerowany
 57.77  Klient nr  14  byl obslugiwany  1.04
 64.54  Klient nr  15  wygenerowany
 67.66  Klient nr  15  byl obslugiwany  3.12
 70.38  Klient nr  16  wygenerowany
 71.76  Klient nr  17  wygenerowany
 73.14  Klient nr  16  byl obslugiwany  2.76
 74.69  Klient nr  18  wygenerowany
 74.84  Klient nr  17  byl obslugiwany  3.08
 75.27  Klient nr  19  wygenerowany
 77.14  Klient nr  18  byl obslugiwany  2.44
 77.27  Klient nr  19  byl obslugiwany  2.00
 81.14  Klient nr  20  wygenerowany
 82.12  Klient nr  21  wygenerowany
 82.45  Klient nr  22  wygenerowany
 84.49  Klient nr  23  wygenerowany
 84.51  Klient nr  20  byl obslugiwany  3.36
 84.67  Klient nr  21  byl obslugiwany  2.54
 87.41  Klient nr  22  byl obslugiwany  4.96
 91.02  Klient nr  23  byl obslugiwany  6.53
 93.25  Klient nr  24  wygenerowany
 94.24  Klient nr  24  byl obslugiwany  0.99
```

```
102.86  Klient nr  25  wygenerowany
103.50  Klient nr  25  byl obslugiwany  0.64
111.15  Klient nr  26  wygenerowany
113.01  Klient nr  26  byl obslugiwany  1.86
113.71  Klient nr  27  wygenerowany
114.04  Klient nr  28  wygenerowany
117.20  Klient nr  27  byl obslugiwany  3.48
118.45  Klient nr  28  byl obslugiwany  4.41
Laczny czas bezczynnosci urzadzenia =  67.86

End of LOGLAN-82 program execution

IBM Personal Computer DOS Version  3.10
```

Powyższy program umożliwia określenie (metodą eksperymentów symulacyjnych) jaki powinien być maksymalny dopuszczalny czas obsługi klienta przez urządzenie, aby przy danym strumieniu zgłoszeń (o przyjętym rozkładzie prawdopodobieństwa) maksymalny czas oczekiwania klienta w kolejce nie przekroczył zadanej liczby jednostek czasowych (tzw. progu cierpliwości), względnie liczba klientów w kolejce była zawsze mniejsza niż (np.) 4 . Program ten można rozszerzyć na większą liczbę rodzajów urządzeń i wiele kolejek wzajemnie ze sobą powiązanych. Urządzenia mogą być ze sobą powiązane w warstwie technicznej, kolejki natomiast w warstwie informacyjnej. Urządzeniami mogą być np. kasy biletowe, informatory dla podróżnych, automaty do rozmiany pieniędzy, itp.

4. Symulacja systemu o zasobach dzielonych

Rozważmy system składający się z kilku równolegle wykonywanych procesów (każdy na innym procesorze). Procesy synchronizowane są za pomocą monitora. Użycie zasobów dzielonych zadeklarowanych w monitorze musi być poprzedzone zadaniem dostępu do monitora (procedura MONITOR_ENTRY). W danej chwili co najwyżej jeden proces może mieć dostęp do monitora. Gdy proces zwalnia monitor, wywoływana jest procedura MONITOR_EXIT. Przykład niniejszy jest opisany wstępnie w publikacji [6]. Natomiast zagadnienia symulacji systemów liczących są omówione w [5].

Poniżej podano listing programu symulacyjnego realizującego przedstawiony problem:

```
PROGRAM ZASD; (* Symulacja systemu o zasobach dzielonych [6] *)
(*$L-*)
(*..... tresc klasy SIMULATION *)
(*$L+*)

BEGIN
 PREF SIMULATION BLOCK;

 UNIT PARPROCESS: SIMPROCESS CLASS(NR: INTEGER),
 (* ... proces wykonywany rownolegle ... *)
```

```
  BEGIN
   DO
    WRITELN("Lokalne akcje procesu numer ",NR:3);
    ;;;;;;;;;;;;;;;;;;
    CALL HOLD(RANDOM);
    CALL MONITOR_ENTRY;
    WRITELN
    ("Proces numer ",NR:3," dziala na zasobach dzielonych");
    ;;;;;;;;;;;;;;;;;;
    CALL HOLD(RANDOM);
    CALL MONITOR_EXIT;
   OD;
  END PARPROCESS;

 UNIT MONITOR_ENTRY: PROCEDURE;
 (* Procedura wejscia *)
   BEGIN
    IF SEMAFOR THEN
      (* ... monitor jest zajety przez inny proces ... *)
      CALL CURRENT.INTO(QUEUE);
      CALL PASSIVATE;
    FI;
    (* ... proces zajmuje monitor ... *)
    SEMAFOR:=TRUE;
   END MONITOR_ENTRY;

 UNIT MONITOR_EXIT: PROCEDURE;
 (* Procedura wyjscia *)
   VAR PROC: PARPROCESS;
   BEGIN
     SEMAFOR:=FALSE; (* ... zwolnienie monitora ... *)
     IF NOT QUEUE.EMPTY THEN
       (* ... uaktywnienie pierwszego procesu   ... *)
       (* ... czekajacego w kolejce do monitora ... *)
       PROC:=QUEUE.FIRST;
       CALL QUEUE.OUTFIRST;
       CALL RUN(PROC);
     FI;
   END MONITOR_EXIT;

 VAR (* Zmienne globalne *)
   SIMULATION_PERIOD: REAL,     (* Czas symulacji       *)
   SEMAFOR          : BOOLEAN, (* Semafor do monitora *)
   QUEUE            : FIFO,    (* Kolejka do monitora *)
   I                : INTEGER;

 CONST N=10; (* Przyjeta liczba symulowanych procesow *)

 BEGIN (* Program glowny *)
  QUEUE:=NEW FIFO;
  SIMULATION_PERIOD:=4.2;
  FOR I:=1 TO N DO CALL SCHEDULE(NEW PARPROCESS(I),TIME) OD;
  WRITELN("Start symulacji ...");
  CALL HOLD(SIMULATION_PERIOD);
  WRITELN("... Koniec symulacji");
 END;

END ZASD.
```

Przykładowy przebieg wykonania programu ZASD jest następujący:

```
IIUW LOGLAN-82  Concurrent Executor  Version 4.35
May 21, 1988
(C)Copyright  Institute of Informatics, University of Warsaw
```

```
Start symulacji ...
Lokalne akcje procesu numer   3
Lokalne akcje procesu numer   9
Lokalne akcje procesu numer   4
Lokalne akcje procesu numer   2
Lokalne akcje procesu numer   8
Lokalne akcje procesu numer   7
Lokalne akcje procesu numer   6
Lokalne akcje procesu numer  10
Lokalne akcje procesu numer   5
Lokalne akcje procesu numer   1
Proces numer   4  działa na zasobach dzielonych
Proces numer   2  działa na zasobach dzielonych
Lokalne akcje procesu numer   4
Proces numer   3  działa na zasobach dzielonych
Lokalne akcje procesu numer   2
Proces numer   1  działa na zasobach dzielonych
Lokalne akcje procesu numer   3
Proces numer   9  działa na zasobach dzielonych
Lokalne akcje procesu numer   1
Proces numer   8  działa na zasobach dzielonych
Lokalne akcje procesu numer   9
Proces numer  10  działa na zasobach dzielonych
Lokalne akcje procesu numer   8
Proces numer   5  działa na zasobach dzielonych
Lokalne akcje procesu numer  10
... Koniec symulacji

End of LOGLAN-82 program execution
```

Program ZASD może być wykorzystany do badania zagadnień synchronizacji procesów w komputerze wieloprocesorowym. Program ten może być poddany dalszej rozbudowie polegającej na: dynamicznie zmieniającej się liczbie współpracujących procesów, bardziej rozbudowanych procedurach wejścia i wyjścia, względnie większej liczbie monitorów.

5. Symulacja przejazdu grupy pojazdów przez obszar kontrolowany

Innym przykładem wykorzystania walorów symulacyjnych języka LOGLAN może być przejazd grupy abstrakcyjnych pojazdów przez pewien kontrolowany obszar. Przyjmijmy, że pojazdy wyposażone są w napęd spalinowo-elektryczny, oraz że obszarem kontrolowanym jest teren uzdrowiska. Istota sytuacji konfliktowej polega na tym, że pojazdy jadą z zasady z włączonym napędem spalinowym a układ komunikacyjny dróg wymusza przejazd przez teren uzdrowiska, w którym z natury rzeczy panują podwyższone rygory ochrony środowiska (m.in. przed spalinami). Kontrolę poziomu spalin w otoczeniu prowadzi w sposób ciągły Centrum Zarządzania Uzdrowiskiem. Jeżeli poziom spalin przekroczy wartość dopuszczalną, wówczas Centrum wysyła drogą radiową komunikat nakazujący wszystkim pojazdom przełączenie napędu na elektryczny. Niestety, nie każdy pojazd respektuje polecenie Centrum; są

tego dwa powody: niewłączenie odbiornika radiowego na daną częstotliwość lub zwykłe niezdyscyplinowanie. Centrum natomiast dysponuje możliwością sankcji w postaci zatrzymania (na czas nieokreślony) niezdyscyplinowanego pojazdu przez lotny patrol.

LISTA ELEMENTÓW: Centrum, Otoczenie (środowisko), Pojazdy ;
LISTA ZMIENNYCH OPISOWYCH:

Centrum:
- SYTUACJA_POJAZDY: INTEGER, (aktualna liczba pojazdów na terenie uzdrowiska)
- SYTUACJA_SPALINY: REAL, (aktualny poziom spalin w otoczeniu)
- KOMUNIKAT : BOOLEAN, (wysłanie komunikatu)

Otoczenie:
- LICZBA_POJAZDÓW : INTEGER,
- SUMA_SPALIN : REAL,

Pojazd:
- CZAS_PRZEJAZDU : REAL,
- SPALINY : REAL,
- NAPĘD_SPALINOWY : BOOLEAN, (włączony, wyłączony)

INTERAKCJA ELEMENTÓW:
- pojazdy oddziaływują na Otoczenie poprzez wydzielanie spalin,
- oddziaływania wzajemne pojazdów (sytuacja ruchowa) nie są przedmiotem rozważań,
- Centrum w sposób ciągły dokonuje pomiarów parametrów otoczenia, w przypadku gdy poziom spalin przekroczył wartość dopuszczalną - wysyłany jest komunikat nakazujący przełączenie napędu oraz dokonywana jest wyrywkowa kontrola i zatrzymywanie niezdyscyplinowanych pojazdów na czas nieokreślony.

Poniżej podano listing programu POJAZDYS symulującego przedstawione powyżej zagadnienie:

```
PROGRAM POJAZDYS;
(* Symulacja przejazdu grupy pojazdow przez obszar kontrolowany *)

(*$L-*)
(*..... tresc klasy SIMULATION *)
(*$L+*)

BEGIN
PREF SIMULATION BLOCK;

VAR  (* zmienne globalne *)
  C: CENTRUM,
  O: OTOCZENIE,
  U: ARRAYOF POJAZD,
  N: INTEGER, CZAS_DOJAZDU, A1,B1: REAL,
  N_MAX          : INTEGER,(* maksymalna liczba pojazdow      *)
  INTENS         : REAL,   (* wspolczynnik intensywn.  spalin *)
```

```
DOP_POZIOM_SPALIN: REAL,    (* dopuszczalny poziom spalin w ot. *)
SR_ZROZUMIENIE   : REAL,    (* wspolczynnik zdyscyplinowania    *)
SR_ROZREG_SILNIKA: REAL,    (* wspolczynnik rozregulow. silnika *)
KROK_REJ         : REAL,    (* krok rejestracji wynikow         *)
CZAS_SYMULACJI   : REAL,
FWE              : FILE,    (* plik dla danych wejsciowych      *)
FWY              : FILE;    (* plik dla wynikow                 *)


UNIT POJAZD: SIMPROCESS CLASS
             ( ZROZUMIENIE, ROZREG_SILNIKA: REAL );
  VAR
    CZAS_PRZEJAZDU : REAL,
    SPALINY        : REAL,
    NAPED_SPALINOWY: BOOLEAN,
    JUZ_JEST       : BOOLEAN;

  BEGIN
    JUZ_JEST:=TRUE;
    CZAS_PRZEJAZDU:=CZAS_SYMULACJI*RANDOM+CZAS_DOJAZDU;
    DO
      IF NOT C.KOMUNIKAT
         THEN NAPED_SPALINOWY:=TRUE; (* z zasady *)
      FI;
      (* losowanie czy POJAZD wykona polecenie Centrum *)
      IF C.KOMUNIKAT AND NAPED_SPALINOWY THEN
        IF RANDOM > SR_ZROZUMIENIE
          THEN (* nie wykonal ... *)
            NAPED_SPALINOWY:=TRUE
          ELSE (* wykonal ...     *)
            NAPED_SPALINOWY:=FALSE;
        FI;
      FI;
      IF NAPED_SPALINOWY
        THEN SPALINY:=INTENS*(RANDOM+ROZREG_SILNIKA)
        ELSE SPALINY:=0.0;
      FI;
      IF TIME>CZAS_PRZEJAZDU
         THEN JUZ_JEST:=FALSE; CALL PASSIVATE;
      FI;
      CALL HOLD(CZAS_PRZEJAZDU*RANDOM);
    OD;
  END POJAZD;


UNIT OTOCZENIE: SIMPROCESS CLASS;
  VAR
    L_POJAZDOW, I: INTEGER,
    SUMA_SPALIN  : REAL;

  BEGIN
    DO
      SUMA_SPALIN, L_POJAZDOW:=0;
      FOR I:=1 TO N_MAX
        DO
          IF U(I).JUZ_JEST THEN
            SUMA_SPALIN:=SUMA_SPALIN+U(I).SPALINY;
            L_POJAZDOW:=L_POJAZDOW+1;
          FI;
        OD;
      CALL HOLD(2); (* praktycznie proces ciagly *)
    OD;
  END OTOCZENIE;


UNIT CENTRUM: SIMPROCESS CLASS;
```

```
  VAR
    SYTUACJA_POJAZDY : INTEGER,
    SYTUACJA_SPALINY : INTEGER,
    KOMUNIKAT        : BOOLEAN,
    NUMER_DO_KONTR   : INTEGER;

  BEGIN

    (* kontrola obszaru nad uzdrowiskiem *)
    DO
      SYTUACJA_POJAZDY:=O.L_POJAZDOW;
      SYTUACJA_SPALINY:=O.SUMA_SPALIN;
      WRITE(FWY,TIME:6:2,"   L.POJ. ",SYTUACJA_POJAZDY:4);
      WRITELN(FWY,"  SPALINY: ",SYTUACJA_SPALINY);
      IF SYTUACJA_SPALINY >= DOP_POZIOM_SPALIN
        THEN KOMUNIKAT:=TRUE ELSE KOMUNIKAT:=FALSE
      FI;
      (* wyrywkowa kontrola *)
      IF KOMUNIKAT THEN
        DO
          NUMER_DO_KONTR:=ROUND(N_MAX*RANDOM);
          IF NUMER_DO_KONTR<LOWER(U) OR NUMER_DO_KONTR>UPPER(U)
            THEN REPEAT
          FI;
          IF U(NUMER_DO_KONTR)=NONE
            THEN EXIT
            ELSE IF U(NUMER_DO_KONTR).NAPED_SPALINOWY
                    AND U(NUMER_DO_KONTR).JUZ_JEST
                  THEN (* zatrzymanie POJAZDU *)
                    U(NUMER_DO_KONTR).JUZ_JEST:=FALSE;
                    CALL CANCEL(U(NUMER_DO_KONTR))
                FI;
            EXIT;
          FI;
        OD;
      FI;
      CALL HOLD(KROK_REJ);
    OD;
  END CENTRUM;


BEGIN (* program glowny *)

  (* czytanie parametrow symulacji *)
  OPEN(FWE,TEXT,UNPACK("DANE.WE")); CALL RESET(FWE);
  READLN(FWE,N_MAX,INTENS,DOP_POZIOM_SPALIN,SR_ZROZUMIENIE,
         SR_ROZREG_SILNIKA,KROK_REJ,CZAS_SYMULACJI);
  KILL(FWE);

  (* echo danych wejsciowych *)
  OPEN(FWY,TEXT,UNPACK("WYNIKI.DAT")); CALL REWRITE(FWY);
  WRITELN(FWY,"=============================================");
  WRITELN(FWY,"   P A R A M E T R Y    S Y M U L A C J I    ");
  WRITELN(FWY,"=============================================");
  WRITELN(FWY,"  N_MAX                 = ",N_MAX);
  WRITELN(FWY,"  INTENS                = ",INTENS);
  WRITELN(FWY,"  DOP_POZIOM_SPALIN     = ",DOP_POZIOM_SPALIN);
  WRITELN(FWY,"  SR_ZROZUMIENIE        = ",SR_ZROZUMIENIE);
  WRITELN(FWY,"  SR_ROZREG_SILNIKA     = ",SR_ROZREG_SILNIKA);
  WRITELN(FWY,"  KROK_REJ              = ",KROK_REJ);
  WRITELN(FWY,"  CZAS_SYMULACJI        = ",CZAS_SYMULACJI);
  WRITELN(FWY,"=============================================");

  (* inicjacja generatora liczb pseudolosowych *)
  CALL RANSET(888.88);
```

```
  (* generowanie grupy pojazdow *)
  ARRAY U DIM(1:N_MAX);
  FOR N:=1 TO N_MAX
    DO
      A1:=(RANDOM+SR_ZROZUMIENIE)/2;
      B1:=(RANDOM+SR_ROZREG_SILNIKA)/2;
      U(N):=NEW POJAZD(A1,B1);
      (* wstepny czas dojazdu *)
      CZAS_DOJAZDU:=(CZAS_SYMULACJI/10)*RANDOM;
      CALL SCHEDULE(U(N),CZAS_DOJAZDU); U(N).JUZ_JEST:=FALSE;
    OD;

  (* generowanie Centrum oraz Otoczenia *)
  C:=NEW CENTRUM;   CALL SCHEDULE(C,2);
  O:=NEW OTOCZENIE; CALL SCHEDULE(O,1);

  (* karuzela zdarzen ... zainicjowana *)
  CALL HOLD(CZAS_SYMULACJI);
  WRITELN(FWY,"=============================================");
  KILL(FWY);

END
END POJAZDYS.
```

Program POJAZDYS zrealizowany został przy użyciu 3-ch simprocessów: POJAZD, OTOCZENIE i CENTRUM. Simprocess POJAZD realizuje akcje przejeżdżającego pojazdu. Jeżeli nie ma komunikatu Centrum, to włączony jest napęd spalinowy; jeżeli natomiast jest komunikat - wtedy losowane jest *rzeczywiste respektowanie nakazu* i w zależności od jego wyniku (uzależnionego od danej wejściowej SR_ZROZUMIENIE) wyłączany jest napęd spalinowy. Simprocess OTOCZENIE kumuluje sumaryczny poziom spalin oraz zlicza liczbę pojazdów w uzdrowisku. Simprocess CENTRUM w otwartej pętli nieskończonej dokonuje kontroli otoczenia i rejestrowania sytuacji bieżącej. W zależności od "sytuacji w spalinach" zmienna KOMUNIKAT ustawiana jest w stan TRUE lub FALSE. Gdy poziom spalin jest wyższy od dopuszczalnego - dokonywana jest wyrywkowa kontrola numeru pojazdu, który w przypadku nie wyłączonego napędu spalinowego zostaje zatrzymany na czas nieokreślony (instrukcja CALL CANCEL ...) .

Moduł główny programu wczytuje z dysku parametry symulacji, inicjuje generator liczb pseudolosowych, generuje instancje wszystkich simprocessów a następnie przekazuje im sterowanie pośrednio poprzez wykonanie instrukcji CALL HOLD(CZAS_SYMULACJI).

Poniżej podano przykładowe trzy przebiegi wykonania programu POJAZDYS:

przebieg 1

```
=============================================
   P A R A M E T R Y    S Y M U L A C J I
=============================================
  N_MAX                =        5
  INTENS               =        3.0000
  DOP_POZIOM_SPALIN    =       30.0000
  SR_ZROZUMIENIE       =        0.5000
  SR_ROZREG_SILNIKA    =        0.3000
```

```
  KROK_REJ              =       10.0000
  CZAS_SYMULACJI        =      100.0000
================================================
  2.00    L.POJ.    0   SPALINY:        0
 12.00    L.POJ.    5   SPALINY:       14
 22.00    L.POJ.    4   SPALINY:       11
 32.00    L.POJ.    3   SPALINY:        9
 42.00    L.POJ.    3   SPALINY:        9
 52.00    L.POJ.    3   SPALINY:        9
 62.00    L.POJ.    2   SPALINY:        4
 72.00    L.POJ.    2   SPALINY:        4
 82.00    L.POJ.    2   SPALINY:        4
 92.00    L.POJ.    2   SPALINY:        4
================================================
```

przebieg 2

```
================================================
   P A R A M E T R Y     S Y M U L A C J I
================================================
  N_MAX                 =       5
  INTENS                =       3.0000
  DOP_POZIOM_SPALIN     =      30.0000
  SR_ZROZUMIENIE        =       0.5000
  SR_ROZREG_SILNIKA     =       0.3000
  KROK_REJ              =      10.0000
  CZAS_SYMULACJI        =     100.0000
================================================
  2.00    L.POJ.    1   SPALINY:        3
 12.00    L.POJ.    5   SPALINY:       10
 22.00    L.POJ.    5   SPALINY:       12
 32.00    L.POJ.    5   SPALINY:       12
 42.00    L.POJ.    4   SPALINY:        9
 52.00    L.POJ.    4   SPALINY:        9
 62.00    L.POJ.    4   SPALINY:       11
 72.00    L.POJ.    2   SPALINY:        6
 82.00    L.POJ.    1   SPALINY:        2
 92.00    L.POJ.    1   SPALINY:        2
================================================
```

przebieg 3

```
================================================
   P A R A M E T R Y     S Y M U L A C J I
================================================
  N_MAX                 =      10
  INTENS                =      33.0000
  DOP_POZIOM_SPALIN     =      30.0000
  SR_ZROZUMIENIE        =       0.5000
  SR_ROZREG_SILNIKA     =       0.3000
  KROK_REJ              =      10.0000
  CZAS_SYMULACJI        =     100.0000
================================================
  2.00    L.POJ.    1   SPALINY:       13
 12.00    L.POJ.    8   SPALINY:      210
 22.00    L.POJ.    8   SPALINY:      214
 32.00    L.POJ.    6   SPALINY:      147
 42.00    L.POJ.    6   SPALINY:      147
 52.00    L.POJ.    3   SPALINY:       98
 62.00    L.POJ.    3   SPALINY:       64
 72.00    L.POJ.    3   SPALINY:       64
```

```
 82.00   L.POJ.     2  SPALINY:      22
 82.00   L.POJ.     2  SPALINY:      22
==========================================
```

Wyniki przebiegów 1 i 2 różnią się mimo tych samych danych wejściowych, świadczy to dobrze o jakości loglanowskiego generatora liczb pseudolosowych (pozwala to uzyskać autentyczny niedeterminizm w przeprowadzanych seriach eksperymentów symulacyjnych - co nie jest obojętne dla badań o charakterze statystycznym). W przypadku przebiegu nr 3 widać sukcesywne zmniejszanie liczby pojazdów będących w ruchu na skutek działalności Centrum w sytuacji, gdy poziom spalin przekroczył dopuszczalną wartość.

6. Uwagi końcowe

Na podstawie zaprezentowanych w artykule przykładów programów, a także na podstawie informacji zawartych w [1] i [2], można stwierdzić dużą przydatność LOGLANu do budowy programów symulacyjnych dla różnego rodzaju zastosowań. Należy w tym miejscu podkreślić dodatkowy fakt, że loglanowska klasa SIMULATION może być rozbudowana o nowe procedury (dalsze generatory liczb pseudolosowych, histogramy, wykresy wyników, itd.), a także o elementy symulacji mieszanej (dyskretno-ciągłej) lub wielokomputerowej. Badania nad tymi zagadnieniami są prowadzone aktualnie w Instytucie Transportu Politechniki Śląskiej.

LITERATURA

[1] KONIECZNY R.: Zasoby symulacyjne języka LOGLAN - *(niniejszy zeszyt)*

[2] KONIECZNY R.: Język programowania LOGLAN jako narzędzie opisu modelu systemu transportowego - *(niniejszy zeszyt)*

[3] KONIECZNY R., KRAWIEC S.: Wybrane zagadnienia opisu formalnego systemu złożonego - *(niniejszy zeszyt)*

[4] KONIECZNY R.: Specyfikacja formalna w projektowaniu programów komputerowych - *(niniejszy zeszyt)*

[5] MADEY J.: Oprogramowanie wspomagające modelowanie systemów liczących - Projekt OS Kit - PAN Instytut Informatyki, PWN, Warszawa 1980.

[6] SZCZEPAŃSKA D.: Narzędzia symulacyjne artykuł z tomu: *Język programowania LOGLAN* - Materiały Jesiennej Szkoły PTI, Serock 1985

[7] ZEIGLER B.P.: Teoria modelowania i symulacji. PWN, Warszawa 1984.

EXAMPLES OF SOLVING SIMULATION PROBLEMS IN LOGLAN LANGUAGE

Summary

The present article is a continuation of the elaboration [1] referring to the LOGLAN simulation resources. Presented examples of programs can make a real system simulators construction scheme.

The examples of simulation considered in the paper are: an abstract machine operation, system with divided resources and vehicle group overpass throught controlled zone.

On the basis of the program examples presented in the paper one can ascertain great usefulness of LOGLAN for constructing simulation programs for different applications. It should be additionaly stressed that the LOGLAN SIMULATION class may be developed by new procedures (next pseudo-random number generators, bar charts, diagrams of the results, etc.) and also by the elements of mixed (discrete - continuous) simulation or multicomputer simulation.

The research on these problems are carried on in the Institute of Transport of Silesian Technical University.

LÖSUNGSBEISPIELE VON SIMULATIONSPROBLEMEN IN DER SPRACHE LOGLAN

Zusammenfassung

Vorliegender Aufsatz bildet eine Weiterführung der die Simulationsvorräte der Sprache LOGLAN betreffenden Bearbeitung [1]. Die vorgestellten Programmmbeispiele können ein gewisser Schema zum Aufban von Simulatoren der realen Systemen bilden.

Im Aufsatz werden folgende Simulationsbeispiele betrachtet: Bedienung einer abstrakten Einrichtung, System mit aufgeteilten Vorräten sowie Durchfahrt einer Fahrzeugsgruppe durch kontroliertes Gebiet.

Auf Basis präsentierten Beispiele kann man feststellten, daß die Sprache LOGLAN für Aufbau von Simulationsprogrammen bei unterschiedlichen Anwendungen nutzlich ist. Man soll dabei eine zusätzliche Tatsache unterstreichen, daß die LOGLAN - klasse SIMULATION um neu Prozeduren (weitere Pseudozufallszahlgeneratoren, Histograme, Ergebnisdiagramme, u. dsgl. m.) ertweitert werden kann.

Diese klasse kann auch um Elemente der gemischten (diskret-stetigen) und Mehrrechnersimulation ausgebaut werden.

Auf diesen Gebiet laufen aktuell im Transportinstitut der Schlesischen

Technischen Universität Forschungsarbeiten.

ПРИМЕРЫ РЕШЕНИЯ СИМУЛЯЦИОННЫХ ПРОБЛЕМ НА ЯЗЫКЕ ЛОГЛАН

Резюме

Данная работа является продолжением разработки [1], касающейся симуляционных ресурсов языка LOGLAN. Представленные примеры программ могут служит схемой постройки симуляторов реальных систем.

Примерами симуляций, рассматриваемыми в статье, являются: обслуживание мнимого устройства, система с делимыми ресурсами а также проезд некоторого количества транспорта через контралированный участок.

На примере представленных в статье примеров программ, можно видет большую пригодность LOGLANa для получения симуляционных программ для различного рода применемий. Необходимо подчеркнуть, что логлановский класс SIMULATION может быть расширен на новые процедуры (генераторы псевдослучайных чисел, гистограммы, рисунки результов итп.), а также на элементы смешанной симуляции (дискретно-непрерывной) или-же многокомпьютерной. Исследования по рассматриваемой проблематике ведутся в настоящее время в Институте Транспорта Силезского Политехнического Института.