ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 199
Seria: TRANSPORT z. 16 Nr kol. 110

Roman KONIECZNY

ZAGADNIENIE REALIZACJI ŁOGLANOWSKIEGO MODUŁU SIMULATION NA BAZIE JĘZYKA TURBO PASCAL

Streszczenie. W artykule rozważono zagadnienie realizacji odpowiednika łoglanowskiego modułu SIMULATION na bazie języka Turbo PASCAL. Wyspecyfikowane zostały elementy, jakie powinien zawierać moduł umożliwiający budowę programów symulacyjnych. W szczególności rozważono: zagadnienie zastępczej realizacji procesu w języku Turbo PASCAL (wersja 5.0), rekord stanu oraz akcje procesu; zagadnienie sterowania procesami oraz zagadnienie reprezentacji osi czasu, dla którego przedstawiono dziewięć wariantów implementacyjnych. Zwrócono również uwagę na aspekt struktury programu symulacyjnego.

Artykuł stanowi materiał wyjściowy do celów analizy przedprojektowej budowy modułu SIMULATION dla potrzeb symulacji systemów dyskretnych zdarzeń w języku Turbo PASCAL.

1. Wprowadzenie

W ciągu ostatnich lat można zaobserwować bardzo szybki rozwój języka Turbo PASCAL - licencjonowanego przez amerykańską firmę Borland International Co [1,2]. Język ten posiada zintegrowane środowisko programowe (edytor, kompilator, linker, debugger), możliwość rozłącznej kompilacji modułów, bogatą bibliotekę procedur pomocniczych oraz możliwość programowania obiektowego.

Przedstawione zalety skłaniają do rozważenia kwestii budowy na bazie Turbo PASCAL-a uniwersalnego modułu (UNIT-u SIMULATION), ułatwiającego symulację systemów dyskretnych zdarzeń. UNIT taki już został, co prawda, zrealizowany w języku LOGLAN [3,4,5,6,7], jednakże efektywność translatora Turbo PASCAL-a jest wielokrotnie wyższa od LOGLAN-u. Ponadto Turbo PASCAL powoli staje się standardem.

Mimo podobieństwa składniowego LOGALN-u i PASCAL-a, bezpośrednie przeniesienie łoglanowskiej klasy SIMULATION na język Turbo PASCAL nie wchodzi w rachubę z uwagi na brak w Turbo PASCAL-u mechanizmu współprogramów oraz ze względu na zbyt wyrafinowane prefiksowanie łoglanowskie. Można jednakże skonstruować skuteczne równoważne mechanizmy zastępcze. Dowodem tego może być moduł MT (multitasking) realizujący wielozadaniowość w Turbo PASCALu [8]. Moduł ten ma jednak inną specyfikę, niż łoglanowska klasa SIMULATION (wzorowana na SIMULI-67) i dla [illegible]

symulacji musiałby być jeszcze rozbudowany (głównie o procedurę postępu czasu).

Podstawowe zasoby, jakie powinien zawierać UNIT SIMULATION, są następujące:

- procedura organizująca symulacyjną oś czasu,
- procedury sterowania procesami [4,7],
- procedury organizujące kolejki proste oraz priorytetowe [4,7],
- procedury generatorów liczb pseudolosowych,
- procedury histogramów,
- procedury dodatkowe związane z obsługą wejścia-wyjścia.

2. Zagadnienie zastępczej realizacji procesu w języku Turbo PASCAL w.5.0

Podstawowym obiektem w programie symulacyjnym jest "proces". Proces można rozumieć jako pewną procedurę "otwartą" imitującą zachowanie procesu rzeczywistego. Proces ma swe atrybuty (zmienne stanu) oraz parametry. Można przyjąć, że zmiana wartości zmiennych stanu procesu (zmiana stanu procesu) odbywa się w pewnych określonych dyskretnych chwilach czasowych. (Zagadnienia związane z opisem formalnym systemu dyskretnych zdarzeń zostały omówione w pracy [9]).

Ogólnie można przyjąć, że proces P jest następującą dwójką:

$$P = \langle ZZS, FPS \rangle$$

gdzie:

ZZS - jest zbiorem wszystkich zmiennych stanu procesu,

FPS - jest zbiorem wszystkich funkcji przejścia stanu.

Funkcje przejścia stanu procesu operują na zbiorze ZSS, a także na wybranych zmiennych stanu innych procesów (i odwrotnie: inne procesy mogą operować na zbiorze ZZS danego procesu). FPS implicite zawierają również strukturę powiązań z otoczeniem. Programistycznie rzecz biorąc, ZZS będzie rekordem, natomiast FPS będą instrukcjami realizującymi poszczególne akcje procesu. Wykonanie każdej akcji prowadzi do zmiany stanu procesu.

Język Turbo PASCAL nie posiada mechanizmów procesów (jak MODULA-2) czy simprocesów (jak LOGLAN). Zastępczy model procesu zrealizowano dla Turbo PASCAL-a w module MT [8], jednakże dotyczy on tam specyfiki wielozadaniowości (w ogólności) i sięga zbyt głęboko do hardware'u komputera (IBM PC). Dlatego w artykule niniejszym przyjęto bardziej konwencjonalne rozwiązanie w sensie pascalowskim i mniej zależne od konkretnej maszyny.

2.1. Rekord stanu procesu

Rekordem stanu procesu można nazwać następującą dwójkę:

$$RS = \langle ZZS^*, ZZP \rangle$$

gdzie:

ZZS^* - jest zbiorem zmiennych stanu oraz parametrów,

ZZP - jest zbiorem zmiennych pomocniczych.

ZZP jest w pewnym sensie "narzutem" informatycznym uzależnionym od przyjętego języka programowania (w jego skład mogą wchodzić np. dodatkowe zmienne wskaźnikowe, zmienne indeksowe, zmienne sterujące w pętlach, itd.).

Przykładowa struktura rekordu stanu procesu może być następująca:

```
type
 Rekord_Stanu_Procesu_Nr1 = record
                            X: -----  }  zbiór deklaracji zmiennych stanu
                            Y: -----  }  oraz paramatrów (ZSS*)
                            -------   }
                            -------   }
                            I: -----  }
                            J: -----  }
                            -------   }  zbiór deklaracji zmiennych
                            -------   }  pomocniczych (ZPP)
                            E: integer}
                            -------   }
                            -------   }
                            end;
```

```
  RS1 = ^Rekord_Stanu_Procesu_Nr1;
```

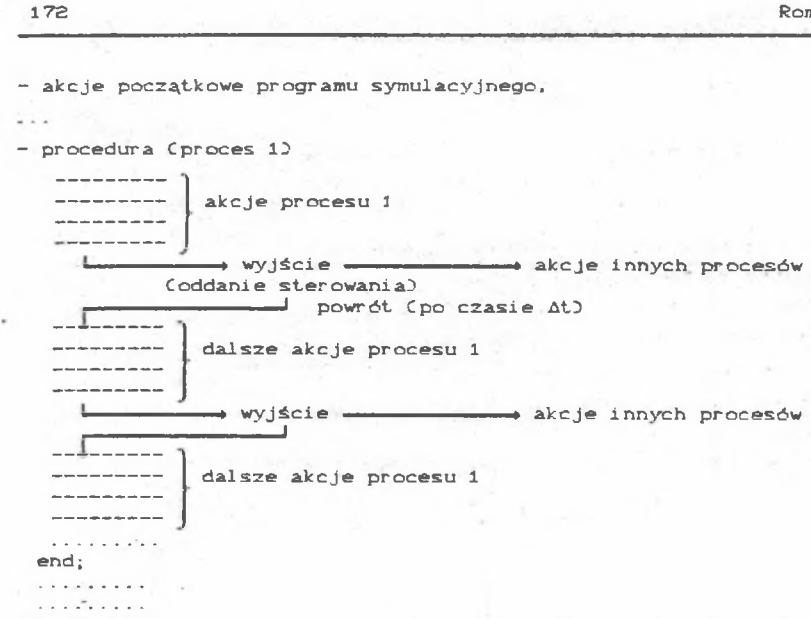
(E - oznacza etykietę kontynuacji procesu).

Rekordy stanu procesów innego typu można również utworzyć według powyższego schematu. Tworzenie kolejnych egzemplarzy procesów danego typu polega na wygenerowaniu odpowiednich rekordów stanu (procedurą NEW). Usunięcie egzemplarza procesu odbywa się poprzez użycie procedury dealokacji (DISPOSE).

2.2. Akcje procesu

Przyjmując, że proces w PASCAL-u będzie zastępczo realizowany w postaci procedury "otwartej", należy rozważyć, w jaki sposób będą wykonywane akcje (instrukcje) tego procesu i jak zapewniona będzie owa "otwartość", czyli wyjście oraz powrót do procedury w określone miejsce (analogicznie jak we współprogramie) celem jej kontynuacji.

Traktując proces najbardziej trywialnie, jako sekwencję instrukcji, można zauważyć, że odpowiednia procedura realizująca akcje procesu (i związane z określonym rekordem stanu) pracować będzie w tzw. przeplocie według schematu:



W programie symulacyjnym przed wyjściem z procesu określa się odstęp czasowy Δt (czasu symulacyjnego) do następnej akcji procesu. Mimo że możliwe (i stosowane) jest losowanie czasu Δt, postępowanie takie ma cechy determinizmu zdarzeń, ponieważ z góry wiadomy jest czas zajścia następnego zdarzenia (aktywacji kolejnego procesu). Określanie odstępów międzyzdarzeniowych nazywane jest w symulacji planowaniem zdarzeń (czyli układaniem kolejności przyszłych zdarzeń na osi czasu symulacyjnego). Kolejność przyszłych zdarzeń zwykle jest przypadkowa, aczkolwiek możliwe jest "wymuszenie" wystąpienia pewnych zdarzeń przed (lub po) innym. Może też być przyjęta taktyka nastawiania czasów Δt na "nieskończoność" (Δt=∞), co oznacza bieżące zawieszanie procesów celem ich późniejszego uaktywnienia w bliżej nieokreślonym czasie.

Warto również w tym miejscu zwrócić uwagę na różnicę w obsłudze klasycznej wielozadaniowości lub współbieżności (np. w systemach operacyjnych) a postępem czasu w symulacji. Przy współbieżności nie są znane czasy wystąpienia kolejnych zdarzeń. Aktywacja procesów odbywa się zazwyczaj cyklicznie poprzez przydział "kwantu" czasu procesora kolejnym procesom. Kwanty czasu, w których procesy są aktywne (praktycznie jeden "bieżący" proces), są odmierzane przez układ przerwań przy wykorzystaniu zegara maszynowego.

W symulacji procesor jest "asynchronicznie" obciążony procesami, a przy obsłudze współbieżności "synchronicznie". Z punktu "widzenia" systemu operacyjnego cały program symulacyjny ze swoimi symulacyjnymi procesami jest także procesem (a raczej należałoby powiedzieć "zadaniem", jednym z wielu, jeżeli system jest wielozadaniowy). W LOGLAN-ie bardzo słusznie rozróżnia się pojęcia "proces" i "simproces" (proces symulacyjny).

Praktycznie rzecz biorąc, w symulacji może zdarzyć się sytuacja, że kolejne akcje bieżącego procesu pochłoną nieskończony czas procesora (np. możliwe jest zapętlenie się któregoś procesu symulacyjnego i nie ma możliwości "samoodwieszenia"), podczas gdy przy pracy wielozadaniowej odwieszanie takie odbywa się automatycznie poprzez układ przerwań. W pracy wielozadaniowej zapętlenie się jednego procesu (zadania) nie może wstrzymać akcji innych procesów (zadań). W symulacji, niestety jest to możliwe, zapętlenie jednego procesu zatrzyma cały program symulacyjny.

Dla pewnej grupy zastosowań możliwe jest prowadzenie symulacji przy wykorzystaniu zegara maszynowego. Częściowe "podparcie się" zegarem maszynowym jest stosowane również, gdy wymagana jest wizualizacja wyników symulacji w czasie rzeczywistym.

Powrót do procedury procesu w miejsce kontynuacji, w takich językach, jak: LOGLAN, MODULA-2, ADA, Concurrent PASCAL, czy nawet C jest organizowany automatycznie. W Turbo PASCAL-u można to zrealizować w sposób konwencjonalny za pomocą mechanizmu skoku i etykiety kontynuacji.

Poniżej podano przykładowy pascalowski model procedury procesu (simprocesu):

```
procedure Proces_Nr1(w:RS1);
 const et1_=1;et2_=2;et3_=3;
 label et1,et2,et3; {etykiety kontynuacji}
 begin
  with w^ do
   begin
    case E of {wymóg techniczny PASCALa}
     et1_: goto et1;
     et2_: goto et2;
     et3_: goto et3;
     else begin {błąd kontynuacji w Procesie Nr1}; exit; end;
    end; {case}
    et1: {...}                                    | zmiana stanu
         {początkowe akcje procesu};
         {...}
         E:=et2_; exit;                           | wyjście
    et2: {...}                                    | zmiana stanu
         {dalsze akcje procesu};
         {...}
         E:=et3_;exit;                            | wyjście
    et3: {...}                                    | zmiana stanu
         {końcowe akcje procesu};
         {...}
         E:=et1_;exit;                            | wyjście
   end;
 end; {Proces_Nr1}
```

W przedstawionym modelu zastępczym charakterystyczne jest użycie instrukcji skoku bezwarunkowego (GOTO) i związanych z tym etykiet kontynuacji. Jest to jedyny wyjątek (z przyczyn technicznych w PASCAL-u), jaki uczyniono dla instrukcji GOTO. (Normalnie instrukcje skoku bezwarunkowego nie powinny być w programie używane). Stałe et1_... związane z etykietami kontynuacji powinny mieć niepowtarzalne wartości.

Aby uniknąć efektów niezamierzonych nie należy przyrywać procesów (wychodzić z procedury procesu) w pętlach FOR. Z uwagi na zastępczy

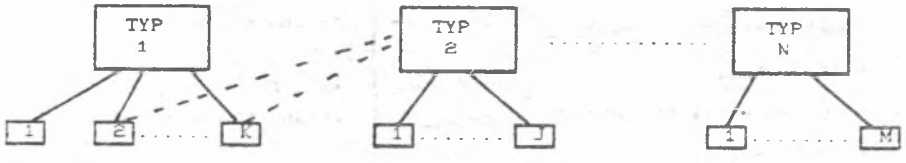
charakter modelu procesu symulacyjnego w PASCAL-u (konieczność stosowania etykiet oraz wstawki case) należy zwrócić szczególną uwagę na czytelność zapisu akcji (instrukcji) procesu i odpowiednie ich komentowanie. Etykiety kontynuacyjne mogą mieć "skojarzeniowe" nazwy, np. START, KONIEC, ROZRUCH, WYBIEG, HAMOWANIE itd.

Stosowanie etykiet kontynuacyjnych, aczkolwiek jest nieco kłopotliwe dla programisty, wymusza jednocześnie na nim pewną zwiększoną dyscyplinę, tj. konieczność dokładnego oznaczania, specyfikowania w nagłówku oraz komentowania "miejsc przerywających" procesu. Dokładne oznaczenie tych miejsc jest niezwykle ważne w programie, ponieważ dzięki temu możliwe jest łatwiejsze odnalezienie "granicznych" instrukcji, gdzie następuje zmiana stanu procesu.

Przed wyjściem z procesu, tj. wykonaniem instrukcji EXIT, należy określić czas międzyzdarzeniowy Δt oraz następną etykietę kontynuacyjną, jeżeli proces ma być kontynuowany. Należy tutaj nadmienić, że w przedstawionym modelu (w odróżnieniu od LOGLAN-u) nie ma żadnych przeciwskazań odnośnie do samodealokacji procesu (tzn. proces może sam siebie usunąć, tj. skasować swój - wybrany dla danego egzemplarza - rekord stanu).

3. Zagadnienie sterowania procesami

Dla każdego wzorca (typu) procesu może istnieć wiele egzemplarzy czynnych procesów. Procesy mogą być ze sobą powiązane. Na rys.1 pokazano istotę pascalowskiego spojrzenia na procesy symulacyjne.



Rys.1. Wzorce i egzemplarze procesów
Fig.1. Masters and copies of processes

Różnica w konstrukcji procesu symulacyjnego (simprocesu) między LOGLAN-em a PASCAL-em polega na tym, że w LOGALN-ie proces (simproces) jest współprogramem i jednocześnie klasą (współprogramem prefiksowanym klasą). Ciało procesu zawiera i wzorzec procedury procesu, i jego rekord stanu. Ponadto nie występują etykiety ani instrukcja GOTO, a kwestia "miejsc przerywających" i kontynuacji jest zautomatyzowana (nie występuje

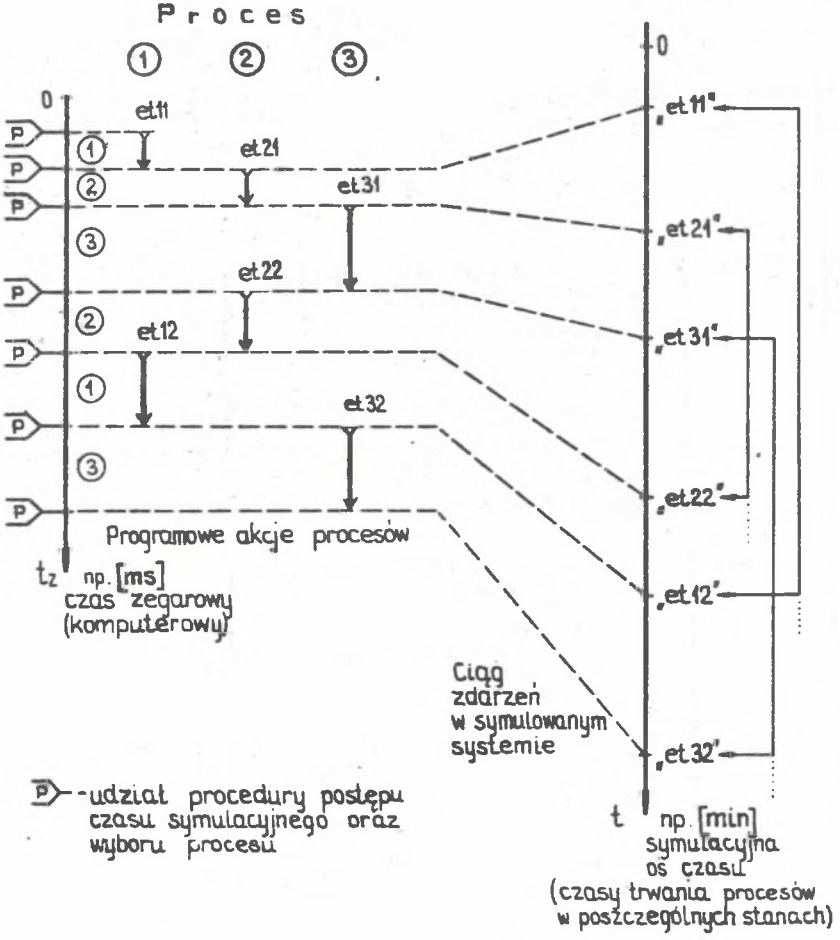
explicite). Jest to ewidentna formalna przewaga LOGLAN-u nad PASCAL-em. (Dlatego w artykule niniejszym mówi się o zastępczej realizacji procesu w PASCAL-u).

Procedura procesu uwzględniać musi wszelkie możliwe wzorce zachowań procesu (wszelkie możliwe zmiany stanów) w różnych sytuacjach. Proces jest powiązany z innymi procesami, jeżeli procedura procesu operuje na rekordach stanu innych procesów (lub jeżeli procedury innych procesów operują na rekordzie stanu wybranego procesu). "Operowanie" na rekordzie stanu dotyczyć może zarówno odczytu, jak i zmiany wartości wybranych zmiennych stanu.

Przedstawiony w p.2 zastępczy uproszczony model procesu można zapisać bardziej szczegółowo:

PROCES

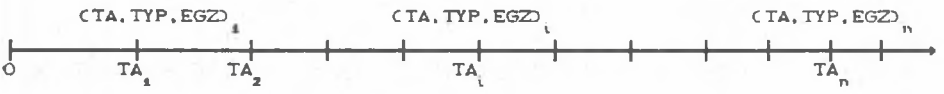
Procesy pracują równolegle (quasi-równolegle). Na rys.2 pokazano zrzutowane na symulacyjną oś czasu czasy trwania przykładowych procesów w różnych stanach oraz czasy obsługi przez komputer akcji poszczególnych procesów.



Rys. 2. Quasi-równoległa praca procesów i symulacyjna oś czasu
Fig. 2. A quasi-parallel work of processes and simulation time axis

Obserwując symulacyjną oś czasu (rys.2), można zauważyć, że np. czas przebywania procesu 1 w stanie 1 przebiega od chwili "et11" do chwili "et12". W symulacji systemów dyskretnych zdarzeń zakłada się, że czas zmiany stanu procesu (tj. czas przejścia ze stanu poprzedniego na następny) jest pomijalnie mały w stosunku do czasu trwania procesu w danym stanie. Klasyczną ilustracją tego problemu może być sygnalizator kolejowy, dla którego istotny jest czas trwania światła zielonego lub czerwonego, a nie czas przejścia ze stanu "czerwone" na "zielone", który jest pomijalny. Z punktu widzenia czasu zegarowego (maszynowego) właśnie te zmiany stanów procesów absorbują procesor. To co na osi czasu symulacyjnego jest pomijalne (akcje procesów), nie jest pomijalne na osi czasu komputerowego. Nie jest również pomijalny czas pracy procedury postępu czasu i wyboru procesu (punkty P rys.2). Ponieważ procedura postępu czasu i wyboru procesu jest najbardziej obciążoną procedurą (pracuje w przeplocie czasowym między akcjami procesów), ważna jest odpowiednia implementacja tej procedury (minimalizacja kosztów czasowych).

Z punktu widzenia procedury postępu czasu na oś czasu symulacyjnego można spojrzeć jeszcze inaczej. Oś ta może być reprezentowana jako uporządkowany rosnąco (według klucza TA) zbiór trójek <TA,TYP,EGZ>, gdzie TA oznacza symulacyjny czas aktywacji procesu, TYP oznacza typ (nazwę, adres) procedury wzorca procesu, a EGZ jest wskazaniem do rekordu stanu danego egzemplarza procesu. Żeby wiedzieć, który proces należy w danej chwili uaktywnić, należy znać najmniejsze (aktualnie pierwsze na osi czsu symulacyjnego) TA, żeby wiedzieć według jakiego wzorca należy wykonać akcje (instrukcje) procesu - należy znać nazwę procedury procesu, żeby wiedzieć od jakiego miejsca we wzorcu należy wykonać akcje (instrukcje) procesu - należy znać wskazanie do jego rekordu stanu. Na rys.3 pokazano ilustrację symulacyjnej osi czasu z punktu widzenia potrzeb programisty.



Rys.3. Oś czasu symulacyjnego z punktu widzenia potrzeb programisty
Fig.3. A simulation time axis from programmer point of view

Chwile czasowe TA należą do zbioru <0,...max INTEGER> lub <0,...,max REAL>. Praktycznie zakresem liczbowym dla TA jest [0,CZAS_SYMULACJI], gdzie wartość zmiennej CZAS_SYMULACJI jest zwykle dużo mniejsza od maksymalnej (na danym sprzęcie) liczby całkowitej lub rzeczywistej. Typy procesów TYP należą do zbioru <TYP_1,...,TYP_maxTYP>. Wskazania EGZ należą do zbioru <1,...,maxEGZ> dla każdego typu. Zmienna TYP odsyła do danego wzorca procedury procesu. Zmienna EGZ jest referencją do wybranego egzemplarza (rekordu stanu) procesu. Referencja ta musi być przekazana procedurze

procesu. Zmienne TA w języku LOGLAN oraz SIMULA noszą nazwę zawiadomień procesów. Oczywiście, tak jak w LOGLAN-ie i SIMUL-i, każdy proces może mieć tylko jedno zawiadomienie na osi czasu.

Sterowanie procesami odbywa się poprzez wykonanie pewnych procedur, które odnoszą się do osi czasu symulacyjnego. Na przykład :

- określenie czasu trwania procesu w danym stanie (loglanowskie HOLD (7)),
- zmiana wartości TA (zawiadomienia) dla danego procesu (loglanowskie SCHEDULE),
- natychmiastowe przekazanie sterowania wybranemu procesowi (loglanowskie RUN),
- zawieszenie wybranego procesu (loglanowskie PASSIVATE oraz CANCEL).

Zrealizowanie wyżej wymienionych procedur w języku Turbo PASCAL w module SIMULATION nie powinno nastręczać specjalnych trudności.

4. Zagadnienie reprezentacji osi czasu

Na osi czasu zachodzą częste zmiany. Pewne zawiadomienia są wstawiane na początek, na koniec lub w środek grupy zawiadomień, inne są usuwane lub przesuwane. Najważniejsze dla procedury postępu czasu jest zidentyfikowanie procesu o najmniejszej wartości zawiadomienia TA i przekazanie mu sterowania (które z kolei może być zwrócone do procedury postępu czasu lub przekazane bezpośrednio innemu procesowi).

Procedura postępu czasu jest najczęściej używana w programie symulacyjnym, dlatego wymaga szczególnego rozważenia ze względu na koszt algorytmu. Możliwy jest tutaj wybór z pewnego wachlarza wariantów: od CSL-owskiego (najgorszego) do loglanowskiego (najlepszego). W wariantach tych oś czasu jest reprezentowana za pomocą tablic, list lub drzew. Poniżej podano niektóre z nich.

WARIANT I a'la CSL

W języku CSL, bazującym na FORTRAN-ie (ze wszystkimi tego negatywnymi konsekwencjami), każdy proces ma swój atrybut czasu. Atrybuty poszczególnych procesów pogrupowane są w tablicach o stałych rozmiarach.

Przyjmując, że n jest liczbą wszystkich czynnych procesów, trzeba przejrzeć wszystkie n atrybutów czasu i określić najmniejszy, następnie wartość tego atrybutu odjąć od wartości wszystkich n atrybutów czasu; z kolei niezależną drogą wyszukiwany jest proces o zerowej wartości atrybutu czasu i jest on uaktywniany. Koszt algorytmu w przypadku pesymistycznym wynosi o(3n).

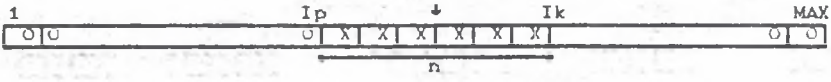
W CSL-u liczba procesów stanowi stałą zadeklarowaną pulę. Wariantu tego nie dotyczą zagadnienia wstawiania na początek, czy też generowania bądź dealokowania procesów. Oś czasu nie występuje explicite. Jest ona "schowana" w atrybutach czasu.

WARIANT II Tablica sortowana

Zakłada się stałą tablicę o wymiarach [1...MAX], w której elementami są trójki <TA,TYP,EGZ>, n jest liczbą aktualnie czynnych procesów (n<MAX). Każde nowe zawiadomienie wstawiane jest na koniec, a następnie tablica jest sortowana według klucza TA, np. przy użyciu standardowej procedury QUICKSORT. Koszt algorytmu wynosi o(n*logn), gdzie logn≡$\log_2 n$.

WARIANT III Tablica ze ześrodkowaniem

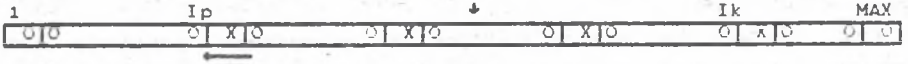
Istotę tego wariantu można zilustrować na następującym schemacie:



Niepuste elementy, tzn. trójki <TA,TYP,EGZ> zgrupowane są w środku tablicy. Ip oznacza indeks elementu pierwszego (pierwszy na osi czasu), Ik - ostatniego. Dostawienie elementu na początek zgrupowania XXX...XXX lub na koniec jest natychmiastowe. Wstawienie wymaga przesunięcia części obecnych już elementów o 1 w prawo lub w lewo. Koszt wyszukania indeksu (metoda podziału połówkowego) wynosi o(logn), koszt przesunięcia (w przypadku pesymistycznym) wynosi o(n/2). Łączny koszt algorytmu: o(logn+n/2). Oczywiście, zawsze musi być spełniony warunek n<MAX, gdzie n jest liczbą czynnych procesów.

WARIANT IV Tablica z rozśrodkowaniem

Różnica w stosunku do wariantu III polega na tym, że zgrupowane elementy zostały rozśrodkowane.



Przeszukiwanie tablicy, celem wstawienia lub usunięcia elementu, odbywa się w zakresie indeksów Ip..Ik. W wariancie tym Ik-Ip jest dużo większe od n. Przyjęcie pewnych odstępów międzyelementowych ma na celu zmniejszenie udziału składnika n/2 (występującego w wariancie III) w ogólnym koszcie algorytmu, ponieważ ewentualnemu przesunięciu ulegnie tylko nieznaczna część elementów, a nie połowa.

Koszt algorytmu dla przypadku pesymistycznego wynosi o(logMAX+K*n), gdzie k jest pewną stałą mniejsza od 0.5.

WARIANT V Lista jednostronna

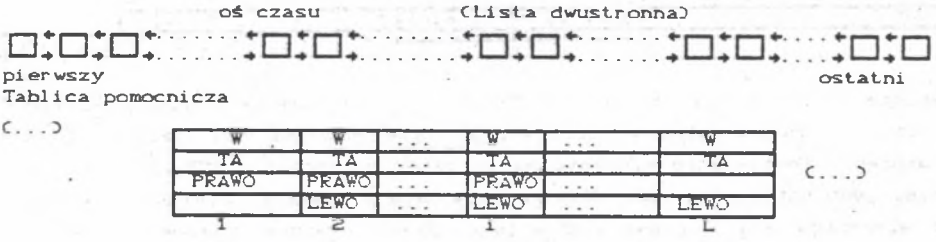
Oś czasu reprezentowana jest w postaci struktury dynamicznej - listy jednostronnej. Jest to klasyczne rozwiązanie, o koszcie algorytmu w przypadku pesymistycznym o(n).

WARIANT VI Lista dwustronna

Oś czasu reprezentowana jest w postaci listy dwustronnej. Jest to również klasyczne rozwiązanie, którego średni koszt algorytmu wynosi o(n/2).

WARIANT VII Lista dwustronna "przyspieszona"

Wadą wariantów V i VI jest konieczność sekwencyjnego przeszukiwania list. Można przyjąć, że operacja ta ulegnie pewnemu przyspieszeniu, jeżeli lista zostanie sprzężona z pomocniczą tablicą wskaźników, jak na poniższym schemacie:

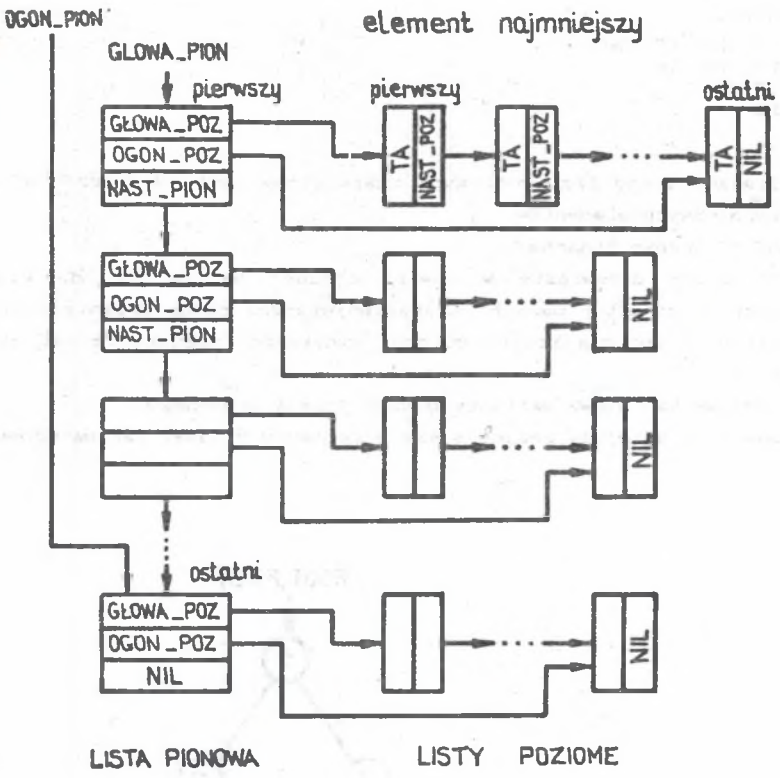


W - wskaźnik do elementu listy,
TA - czas aktywacji procesu,
PRAWO,LEWO - liczby elementów w prawo lub w lewo na liście za elementem, do którego istnieje wskazanie w tablicy pomocniczej.

Tablica pomocnicza jest przeszukiwana według klucza TA, a następnie jest przeszukiwany odpowiedni fragment listy. Optymalny koszt algorytmu wynosi: o(logL+n/2*L).
Aby koszt bieżący był zawsze zbliżony do optymalnego, potrzebne jest okresowe korygowanie tablicy pomocniczej, tak aby na poszczególne wskazanie przypadała mniej więcej ta sama liczba procesów obecnych na liście (tzn. aby wartości zmiennych PRAWO i LEWO niewiele różniły się od siebie). Tablica pomocnicza dla tego wariantu również może być ześrodkowana (analogicznie jak oś czasu w wariancie III).

WARIANT VIII N list jednostronnych

Dalsze przyspieszenie wykonywania operacji na osi czasu można uzyskać poprzez zorganizowanie jej w postaci zbioru N list jednostronnych ("poziomych") jak na poniższym schemacie:



Zbiór list poziomych jest tak zorganizowany, że element o najmniejszej wartości TA (pierwszy na osi czasu) znajduje się zawsze jako pierwszy na pierwszej liście poziomej. Lista "pionowa" (pomocnicza) zawiera wskaźniki do list poziomych (do pierwszego elementu GŁOWA_POZ oraz do ostatniego OGON_POZ).

Po usunięciu elementu najmniejszego układ list wymaga korekcji. Przestawienie list poziomych odbywa się bez naruszenia struktury listy pionowej (następuje tylko przestawienie wskaźników).

Przedstawiona struktura jest w pełni dynamiczna. Nowe elementy mogą być dopisywane tylko na początek lub na koniec odpowiedniej listy poziomej. Wyszukiwanie odpowiedniej listy dla nowego elementu odbywa się "wzdłuż" listy pionowej. W przypadku gdy nowy element nie może być dopisany do żadnej listy - tworzona jest nowa lista pozioma.

Koszt algorytmu dla omawianego wariantu wynosi o(n*logn).

Poniżej podano przykładowy układ liczb egzemplifikujących istotę wariantu zbioru list:

```
  5  7  13  17  99
  8 10  16  18
  9 13  14
 10 12
 11
```

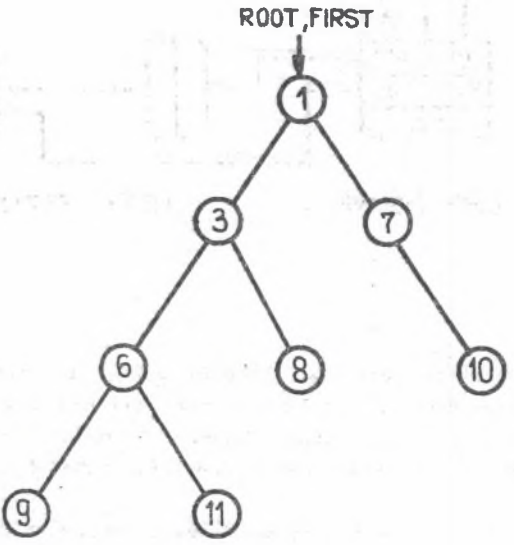
Bieżący układ list poziomych uzależniony jest od "rozrzutu" wartości kolejnych nowych elementów.

WARIANT IX Drzewo binarne

Struktury drzewiaste w chwili obecnej należą już do klasycznych dynamicznych struktur danych. Charakterystyczną cechą drzew binarnych jest logarytmiczna funkcja kosztu o(log n) wynikająca niejako "z definicji" tej struktury.

Możliwe są tu dwa warianty budowy drzewa binarnego:

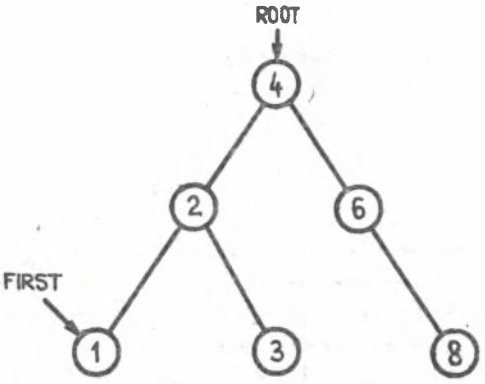
a) element najmniejszy znajduje się w korzeniu drzewa, jak na schemacie:



FIRST – pierwszy na osi czasu

ROOT – korzeń drzewa

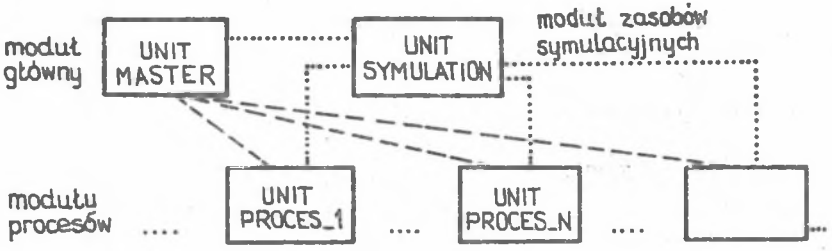
b) element najmniejszy znajduje się w skrajnie lewym węźle, jak na schemacie:



Z punktu widzenia programisty wariant b) jest bardziej logiczny i łatwiejszy w realizacji.

5. Struktura programu symulacyjnego

Struktura programu symulacyjnego w Turbo PASCAL-u jest prosta i przejrzysta. Ilustruje ją rys.4.



Rys.4. Struktura programu symulacyjnego w Turbo PASCAL-u
Fig.4. A simulation program structure in Turbo PASCAL

```
UNIT MASTER jest modułem wiążącym wszystkie inne moduły.
Budowa modułu MASTER może być następująca:
PROGRAM MASTER;
 USES SIMULATION,PROCES_1,...,PROCES_N,...;
 (deklaracje zmiennych globalnych)

 inicjacja symulacji;
 ...
END.
```

Moduł SIMULATION zawiera w sobie procedurę postępu czasu oraz wyboru procesów (szczegóły implementacyjne). Powinien on również udostępniać wszelkie niezbędne do prowadzenia symulacji procedury - wymienione w p.1.

Wybór procesu może być zrealizowany przy wykorzystanie typów proceduralnych lub prościej - tablicy typów procesów - jak poniżej:

```
...
CONST
  max_liczba_typów_procesów = ...;
TYPE
  tablica_typów_procesów = ARRAY[1..max_liczba_typów_procesów] OF
      .RECORD
        typ: INTEGER;
        p  : POINTER;
       END;
 ...
VAR
  procesy: tablica_typów_procesów;
 ...
PROCEDURE wybór_procesu(i:INTEGER);
 BEGIN
  CASE procesy[i].typ OF
   ...
   typ_proces_N:proces_N(procesy[i].p);
   ...
   ELSE niezidentyfikowany proces;
  END; (CASE)
END; (wybór_procesu)
```

Budowa modułu PROCES_N może być następujaca:

```
UNIT PROCES_N
 INTERFACE
 CONST typ_proces_N = ...;(unikalny numer typu procesu)
 (rekord stanu procesu)
 IMPLEMENTATION
 (procedura procesu)
```

```
END.
```

Rekordy stanu poszczególnych procesów wystawione są w sekcji INTERFACE. Oznacza to nieograniczony wzajemny dostęp do atrybutów poszczególnych procesów. Jest to czasem niebezpieczne i nie zawsze czytelne w przypadku powiązań wzajemnych.

Jeżeli procesy wzajemnie sięgają do swoich atrybutów (zmiennych stanu) wskazane jest utworzenie dodatkowej specyfikacji w module MASTER. Przykład zapisu może być następujący:

```
  ...
  atr_proces_N:INTEGER; {← ustawiany w procesie N przez zmienną N
                        → odczytywany w procesie K przez zmienną B}
```

6. Uwagi końcowe

Przestawione w artykule zagadnienia realizacji odpowiednika loglanowskiego modułu SIMULATION na bazie języka Turbo PASCAL musiały być z konieczności naświetlone ogólnie i w pewnym sensie pobieżnie. Szczegóły implementacyjne są bowiem zawsze uzależnione od zespołu programującego. Szereg zagadnień szczegółowych może być rozwiązanych różnorodnie - w zależności od wyników eksperymentów z kompilatorem oraz ze sprzętem. Zawsze jednak w fazie przedprojektowej niezbędna jest analiza wariantów i dyskusja koncepcji.

Przykładowo, dla realizacji procedury postępu czasu, na podstawie wybranych dziewięciu wariantów stwierdzić można, że: warianty tablicowe będą łatwiejsze w realizacji, ale bardziej czasochłonne (kosztowne) i jednocześnie bardziej pamięciochłonne. Warianty z drzewami z kolei są mniej kosztowne i pamięciochłonne, ale wymagają bardziej wyrafinowanych metod programistycznych.

Wybór poszczególnego wariantu jest zawsze kompromisem pomiędzy możliwościami zespołu programującego - z jednej strony, a ograniczeniem czasowym na wykonanie danej implementacji - narzuconym przez zleceniodawcę - z drugiej.

(Zagadnienia realizacji kolejek, generatorów liczb pseudolosowych, histogramów nie zostały w artykule omówione, ponieważ należą one już do klasycznych temetów literatury komputerowej).

LITERATURA

[1] BIELECKI J.: Turbo PASCAL 5.0 - wersja profesjonalna - Wydawnictwa Komunikacji i Łączności, Warszawa 1989.

[2] CIRIC BOBBY, THIES KLAUS DIETER: Turbo PASCAL 5.0/5.5 - te-wi Verlau GmbH 1989.

[3] KONIECZNY R.: Charakterystyka zasobów symulacyjnych języka LOGLAN na potrzeby modelowania systemów transportowych. Zeszyt Naukowy Politechniki Śląskiej, s Transport nr 9, Gliwice 1989.

[4] KONIECZNY R.: Zasoby symulacyjne języka LOGLAN. Zeszyt Naukowy Politechniki Śląskiej, s.Transport nr 13, Gliwice 1989.
[5] KONIECZNY R.: Przykłady rozwiązania problemów symulacyjnych w języku LOGLAN. Zeszyt Naukowy Politechniki Śląskiej, s.Transport nr 13, Gliwice 1989.
[6] KONIECZNY R.: Zagadnienie rozszerzenia loglanowskich zasobów symulacyjnych. Zeszyt Naukowy Politechniki Śląskiej, s.Transport nr 13, Gliwice 1989.
[7] KONIECZNY R.: Język programowania LOGLAN jako narzędzie symulacji systemu transportowego. Zeszyt Naukowy Politechniki Śląskiej, s.Transport nr 13, Gliwice 1989.
[8] TISCHER: Turbo PASCAL intern. Data Becker 1989.
[9] ZEIGLER B.P.: Teoria modelowania i symulacji. PWN, Warszawa 1984.

PROBLEM OF SIMULATION LOGLAN MODULE APPLICATION WHICH IS BASED ON TURBO PASCAL

Summary

Problem of SIMULATION Loglan module application which is based on Turbo PASCAL has been presented in the paper. Module's specified elements which are essential for simulation programmes design have been discussed substituting process realization in Turbo PASCAL (5.0 version) in particular. State record, process actions, process stearing problems, time axis are presentation with 9 implementation alternatives have also been presented in the paper. All this is basic for further pre-design analysis of SIMULATION module for Discrete Events SIMULATION systems in Turbo PASCAL.

REALISIERUNGSPROBLEM DES LOGLANPROGRAMMODULS SIMULATION AUF DER GRUNDLAGE DER PROGRAMMIERSPRACHE TURBO PASCAL

Zusammenfassung

Im Aufsatz wurde das Realisierungsproblem des Gegenstücks zum LOGLAN - Programmodul SIMULATION auf der Grundlage der Programmiersprache TURBO PASCAL betrachtet. Es wurden Elemente spezifiert, die ein den bau von Simulationsprogrammen ermöglichendes Modul beinhalten soll. Es wurde insbesondere betrachtet: Problem der Ersatzrealisierung des Prozesses in der Sprache TURBO PASCAL (Version 5.0), Zustandsrekord und Prozeßaktionen, Problem der Prozeßsteuerung und Representationsproblem der Zeitachse, für

welches neun Implementierungsversionen vorgestellt wurden. Es wurde ebenfalls auf die Struktur des Simulationsprogramms hingewiesen. Der Aufsatz stellt Ausgangsgedankengut der Vorprojektierungsanalyse beim Aufbau des SIMULATION-Programmoduls zum Zwecke der Komputersimulation von Systemen mit dyskreten Ereignissen in der Sprache TURBO PASCAL.

ПРОБЛЕМА РЕАЛИЗАЦИИ ЛОГЛАНОВСКОГО МОДУЛЯ SIMULATTION С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА TURBO PASCAL

Резюме

В статьи рассмотрено проблемы связанные с реализацией эквивалента логлановского модуля SIMULATION на языке Turbo PASCAL. Перечислено составляющие какие должны входит в состав модуля чтоб возможное было получение программ моделирования. Рассмотрено проблему замещающей реализации процесса на языке Turbo PASCAL (вариант 5.0), рекорда состаяния а также действие процесса; проблемы управления процессами, представления оси времени для которой рассмотрено девят вариантов.

Обращено внимание на структуру программ модели. Статья содержит исходный материал для предпроектных анализ строения модуля SIMULATION необходимых при моделировании систем дискретных действии на языке Turbo PASCAL.