ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)

Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2016, № 2 (62)

УДК 510.25:004.422.63.05

В. И. ШИНКАРЕНКО[1*], Г. В. ЗАБУЛА[2*]

[1*]Каф. «Компьютерные информационные технологии», Днепропетровский национальный университет железнодорожного транспорта имени академика В. Лазаряна, ул. Лазаряна, 2, Днепропетровск, Украина, 49010, тел. +38 (056) 373 15 35, эл. почта shinkrenko_vi@ua.fm, ORCID 0000-0001-8738-7225
[2*]Каф. «Компьютерные информационные технологии», Днепропетровский национальный университет железнодорожного транспорта имени академика В. Лазаряна, ул. Лазаряна, 2, Днепропетровск, Украина, 49010, тел. +38 (056) 373 15 35, эл. почта zabulus12@gmail.com, ORCID 0000-0002-8607-5729

КОНСТРУКТИВНАЯ МОДЕЛЬ АДАПТАЦИИ СТРУКТУР ДАННЫХ В ОПЕРАТИВНОЙ ПАМЯТИ: ЧАСТЬ II. КОНСТРУКТОРЫ СЦЕНАРИЕВ И ПРОЦЕССОВ АДАПТАЦИИ

Цель. Вторая часть статьи завершает представление системы конструктивно-продукционных структур (КПС), моделирующих адаптацию структур данных в оперативной памяти (ОП). Цель второй части исследования заключается в разработке модели процесса адаптации данных в ОП, функционирующих в различных программно-аппаратных средах, и сценариев процессов обработки данных. **Методика.** Для осуществления цели применена методология математико-алгоритмического конструктивизма. В данной части работы, на основании обобщенной КПС, путем ее трансформационных преобразований разрабатываются конструкторы сценариев и процессов адаптации. Конструкторы являются интерпретированными, специализированными КПС. Выделяются терминальные алфавиты конструктора сценариев в виде алгоритмов обработки данных и конструктора адаптации – в виде алгоритмических составляющих процесса адаптации. Методика предусматривает разработку правил подстановки, определяющих процесс вывода соответствующих конструкций. **Результаты.** Во второй части статьи представлена часть системы КПС, моделирующей адаптацию размещения данных в ОП, а именно, конструкторов сценариев и процессов адаптации. Результатом реализации конструктора сценариев является набор операций обработки данных в виде текста на языке программирования C#, конструктора процесса адаптации – процесс адаптации, а результатом процесса адаптации – адаптированный бинарный код обработки структур данных. **Научная новизна.** Впервые предложена конструктивная модель процесса обработки данных – сценария, учитывающего порядок и количество обращений к различным элементам структур данных, а также адаптации структур данных к различным программно-аппаратным средам. При этом адаптируется размещение данных в ОП и алгоритмы их обработки. Применение конструктивизма в моделировании позволило в рамках единого подхода и применяемых средств связать модели данных и алгоритмы их обработки с критериями эффективности. Разработанные модели позволяют исследовать процесс адаптации и управлять им. **Практическая значимость.** Разработанные модель и методы позволяют автоматически изменять размещение данных в ОП и их алгоритмические связи в зависимости от эксплуатационных потребностей, конструктивных особенностей аппаратных средств и программной среды функционирования.

Ключевые слова: структура данных; конструктивно-продукционная структура; адаптация; конструктор; преобразователь

Введение

Формальные модели структур данных и алгоритмов их обработки [1−4] разработаны на основе аппарата конструктивно-продукционных структур (КПС) [16]. На основе исследований структур данных [5] и эффективности их размещения в оперативной памяти (ОП) [7, 8, 13-15] и алгоритмов [6, 17], разработана конструктивная модель адаптации данных в ОП.

В первой части статьи [10] дано определение КПС и представлены вспомогательные конструкторы, позволяющие конструировать множество различных вариантов размещения данных в ОП и порождающие соответствующие им программы обработки данных, таких как добавление, поиск и т.д. В этой части работы представлены основные конструкторы сценариев и процессов адаптации.

doi 10.15802/stp2016/67302

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)

Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2016, № 2 (62)

Конструктор сценариев. Моделируемую последовательность операции обработки данных назовем сценарием работы со структурой данных (или просто сценарием).

Уточним, что понимается под операциями обработки данных. Обработка данных выполняется на основе атомарных или примитивных операций обработки. Все примитивные операции обработки СД можно разделить на:

операции не связанные с изменением структуры и значений ее элементов. Такие операции должны быть ограничены исключительно выборкой или поиском значений и исключать другую функциональность, например, определение средних значений по выборке и т. п.;

− операции, связанные с преобразованием структуры данных в том числе: изменение логической структуры (например, замена массива списком, балансировка дерева); добавление/удаление элемента (ов);

− операции, связанные с изменением значений ее элементов без изменения структуры. Такие операции должны быть ограничены исключительно поиском (при необходимости) позиции и заменой значения и исключать преобразование данных;

− операции, связанные с изменением значений ее элементов и связанные с этим изменения структуры. Например, добавление элемента в упорядоченный список.

Формальной структурой для формирования сценариев обработки данных назовем специализированную ОКПС C_{SC}:

$$C =< M, \Sigma, \Lambda >_S \mapsto C_{SC} =< M_{SC}, \Sigma_{SC}, \Lambda_{SC} >,$$

где $\Sigma_{SC} =< \Xi_{SC}, \Theta_{SC}, \Phi_{SC}, \{\rightarrow\} >$, $\Xi_{SC} = \{\circ, ;\}$, $\Lambda_{SC} = \Lambda_{15} \cup \Lambda_{16}$. $\Lambda_{15} = \{M_{SC} \supset (T_{SC} \cup N_{SC}), T_{SC} = \{C_{\#}\}, N_{SC} = \{\alpha_i\}\}$.

Частичная аксиоматика Λ_{16} приведена ниже.

Терминальный алфавит T_{SC} состоит из множества $C_{\#}$ – лексем языка C#.

Операции связывания терминалов и нетерминалов:

− ввод данных из внешней среды $\circ x$, где x – идентификатор введенных данных. Для выполнения операции требуется внешний исполнитель, который предоставляет данные. Внешний исполнитель представляет собой программно-аппаратную среду, возможно управляемую пользователем;

− связывание элементов сценария $x; y$, где x и y – элементы сценария, которые выполняются последовательно.

− Операции над атрибутами:

$\vee(a,b)$ – генерации случайного числа на промежутке $[a,b]$; × – умножения, – – вычитания, = – сравнения на равенство, := – присвоения; $\div(c;t;f)$ – выбора значения как в C_{LD}.

Для интерпретации C_{SC} необходима модель исполнителя, которую представим в виде базовой алгоритмической структуры (БАС):

$$C_{A,SC} = \left\langle M_{A,SC}, V_{A,SC}, \Sigma_{A,SC}, \Lambda_{A,SC} \right\rangle,$$

где $M_{A,SC}$ – неоднородный носитель, $V_{A,SC}$ – множество образующих алгоритмов, $\Sigma_{A,SC}$ – сигнатура и $\Lambda_{A,SC}$ – аксиоматика. Носитель $M_{A,SC} \supset T_{SC} \cup N_{SC} \cup \Omega(C_{A,SC}) \cup W$, где $\Omega(C_{A,SC})$ все сформированные алгоритмами алгоритмической структуры конструкции; W – множество допустимых значений атрибутов. Множество алгоритмов

$V_{A,SC} \supset \{ A_1^0 |_{A_i,A_j}^{A_i \cdot A_j}, A_2^0 |_{Z_1,Z_2,A_i}^{:A_i}, A_3 |_{l_h,l_q,f_i}^{f_j}, A_4 |_{f_i,\Psi}^{f_j}, A_5 |_{f_i,\Psi}^{f_j}, A_6 |_{a}^{b}, A_7 |_{a,b}^{c}, A_8 |_{a,b}^{c}, A_9 |_{a,b}^{c}, A_{10} |_{a,b}^{p}, A_{11} |_{c,f,t}^{v}, A_{12} |_{\Re}^{x}, A_{13} |_{l_i,l_j}^{l_i,l_j} \} \bigcup V_W \subset \Omega(C_{A,SC})$.

$A_1^0 |_{A_i,A_j}^{A_i \cdot A_j}$, $A_2^0 |_{Z_1,Z_2,A_i}^{:A_i}$ такие как в $C_{A,LD}$, $A_3 |_{l_h,l_q,f_i}^{f_j}$ такой как $A_5 |_{l_h,l_q,f_i}^{f_j}$ в $C_{A,LD}$, $A_4 |_{f_i,\Psi}^{f_j}$, такой как $A_6 |_{f_i,\Psi}^{f_j}$ в $C_{A,LD}$, $A_5 |_{f_i,\Psi}^{f_j}$ такой как $A_7 |_{f_i,\Psi}^{f_j}$ в $C_{A,LD}$;

$A_6 |_{a}^{b}$, $A_7 |_{a,b}^{c}$, $A_8 |_{a,b}^{c}$, $A_9 |_{a,b}^{c}$ алгоритмы реализующие операции присваивания, сравнения на равенство, умножения и вычитания соответственно;

$A_{10} |_{a,b}^{p}$ генерация случайного числа из промежутка $[a,b]$;

$A_{11} |_{c,f,t}^{v}$ соответствует $A_8 |_{c,a,b}^{v}$ из $C_{A,LD}$;

doi 10.15802/stp2016/67302

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)

Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2016, № 2 (62)

$A_{12}\,|^{x}_{\Re}$ алгоритм ввода данных, $x \in T_{SC}$, $\Re$ – программно-аппаратная среда;

$A_{13}\,|^{l_i,l_j}_{l_i,l_j}$ алгоритм связывания элементов, $l_i, l_j \in M_{SC}$.

Конструктор сценариев представляет собой совокупность моделей сценариев. Он является интерпретацией формальной структурой C_{SC} алгоритмической структуры $C_{A,SC}$:

$$< C_{SC} =< M_{SC}, \Sigma_{SC}, \Lambda_{SC} >,$$

$$C_{A,SC} =< M_{A,SC}, \Sigma_{A,SC}, \Lambda_{A,SC} >>$$

$$_{I}\mapsto\ {}_{I}C_{SC} =< M_{I,SC}, \Sigma_{I,SC}, \Lambda_{I,SC} >,$$

где $_{I}\mapsto$ – операция интерпретации;

$\Lambda_{I,SC} = \Lambda_{SC} \cup \Lambda_{17}$; $\Lambda_{17} = \{ (A_1^0\,|^{A_i \cdot A_j}_{A_i,A_j} \hookleftarrow \cdot);$

$(A_2^0\,|^{A_i}_{Z_1,Z_2,A_i} \hookleftarrow :);\ (A_3\,|^{f_j}_{l_h,l_q,f_i} \hookleftarrow \Rightarrow);\ (A_4\,|^{f_j}_{f_i,\Psi} \hookleftarrow |\Rightarrow);$

$(A_5\,|^{f_j}_{f_i,\Psi} \hookleftarrow ||\Rightarrow);\ (A_6\,|^{b}_{a}, \hookleftarrow :=);\ (A_7\,|^{c}_{a,b}, \hookleftarrow =);$

$(A_8\,|^{c}_{a,b}, \hookleftarrow \times);\ (A_9\,|^{c}_{a,b}, \hookleftarrow -);\ (A_{10}\,|^{p}_{a,b}, \hookleftarrow \vee);$

$(A_{11}\,|^{v}_{c,f,t}, \hookleftarrow \div);\ (A_{12}\,|^{x}_{\Re}, \hookleftarrow \circ);\ (A_{13}\,|^{l_i,l_j}_{l_i,l_j}, \hookleftarrow ;);\ \}$.

Алгоритм A_{12} может быть реализован как C# generic функция с переменным количеством параметров. Например:

```
public T GetParameter<T>(params object[] inputs)
```

$$_{I}C_{SC} =< M_{I,SC}, \Sigma_{I,SC}, \Lambda_{I,SC} >$$

$$_{K}\mapsto\ C_{SC_{BMP}} =< M_{SC_{BMP}}, \Sigma_{SC_{BMP}}, \Lambda_{SC_{BMP}} >$$

где $M_{SC_{BMP}} \supset N_{SC_{BMP}} \cup T_{SC_{BMP}}$; $\Sigma_{SC_{BMP}} = \Sigma_{I,SC}$; $\Lambda_{SC_{BMP}} = \Lambda_{I,SC} \cup \Lambda_{18} \cup \Lambda_{19}$. $\Lambda_{18} =< T_{SC_{BMP}} = \{ T_{PL} \cup \{ x, y, root, DeleteAt, Resize, bpp, width, height \} \}$, $N_{SC} = \{ SCELEMENT, SCENARIO, \sigma \} >$.

Частичная аксиоматика Λ_{19} заключается в следующем.

Внешним исполнителем задано: интервал количества применяемых правил $N_{\min}, N_{\max}$ и вероятности использования каждого правила $\vec{p} =< p_0, ..., p_k >$, где k – количество операций. На основании чего определяется количество выполнения каждой операции вектором $\vec{n} =< n_0, ..., n_k >$.

Формируемые сценарии обработки BMP файлов в ОП включают следующие операции обработки данных: изменение цвета пикселя по координатам (правило s_3), получение значение цвета по координатам (s_4), установка цвета пикселя белым (s_5), изменение общего размера изображения (s_6), получение ширины изображения (s_7), изменение битности изображения (s_8), изменение ширины изображения (s_9), изменение высоты изображения (s_{10}), получение значения высоты изображения (s_{11}), получение значения битности изображения (s_{12}), изменение компонент R, G, B пикселя по координатам (соответственно компонентам s_{13}, s_{14}, s_{15}).

Правила вывода имеют вид $\psi_i =< s_i, g_i >$. Зададим следующие отношения подстановки:

$$s_0 =< \sigma \to \Omega(C_{PLI}); SCENARIO >$$

$$s_1 =< SCENARIO\ {}_{\vec{d}}\to SCELEMENT;$$

$$SCENARIO >$$

$$s_2 =< SCENARIO\ {}_{\vec{d}}\to SCELEMENT; >$$

$$s_3 =< SCELEMENT\ {}_{\vec{d}}\to \circ x; \circ y; \circ rgb;$$

$$root.ImageData[x, y]\ = rgb; >$$

$$s_4 =< SCELEMENT\ {}_{\vec{d}}\to \circ x; \circ y;$$

$$root.ImageData\big[x,\ y\big]; >$$

$$s_5 =< SCELEMENT\ {}_{\vec{d}}\to \circ x; \circ y;$$

$$root.ImageData.DeleteAt\big(x,\ y\big); >$$

$$s_6 =< SCELEMENT\ {}_{\vec{d}}\to \circ w; \circ h;$$

$$root.ImageData.Resize\big(w,\ h\big); >$$

$$s_7 =< SCELEMENT\ {}_{\vec{d}}\to$$

$$root.ImageHeader.Width; >$$

$$s_8 =< SCELEMENT\ {}_{\vec{d}}\to \circ bpp;$$

doi 10.15802/stp2016/67302

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)

Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2016, № 2 (62)

$$root.ImageHeader.BitsPerPixel = bpp;>$$

$$s_9 =< SCELEMENT\ _{\vec{d}} \rightarrow \circ width;$$

$$root.ImageHeader.Width = width;>$$

$$s_{10} =< SCELEMENT\ _{\vec{d}} \rightarrow \circ height;$$

$$root.ImageHeader.Height = height;>$$

$$s_{11} =< SCELEMENT\ _{\vec{d}} \rightarrow$$

$$root.ImageHeader.Height;>$$

$$s_{12} =< SCELEMENT\ _{\vec{d}} \rightarrow$$

$$root.ImageHeader.BitsPerPixel;>$$

$$s_{13} =< SCELEMENT\ _{\vec{d}} \rightarrow \circ r; \circ x; \circ y;$$

$$root.ImageData[x, y].R = r;>$$

$$s_{14} =< SCELEMENT\ _{\vec{d}} \rightarrow \circ g; \circ x; \circ y;$$

$$root.ImageData[x, y].G = g;>$$

$$s_{15} =< SCELEMENT\ _{\vec{d}} \rightarrow \circ b; \circ x; \circ y;$$

$$root.ImageData[x, y].B = b;>$$

Правила над атрибутами:

$$g_0 =< N = \vee[N_{\min}, N_{\max}]; n_i := N \times p_i >,$$

$g_i =< d_i = \div(n_i = 0,1,0); n_i := n_i - 1 >$, при $i = [3...15]$.

В результате реализации структуры $C_{SC_{BMP}}$ формируется множество сценариев обработки данных BMP файлов. Например, часть сценария:

```
public T GetParameter<T>(params object[] inputs)
{
Console.WriteLine("Enter parameter: {0}", inputs[0]);
Return Convert.To<T>(Console.ReadLine());
}
int height = GetParameter<int>("height");
root.ImageHeader.Height = height;
int r = GetParameter<int>("r");
int x = GetParameter<int>("x");
int y = GetParameter<int>("y");
root.ImageData[x,y].R = r;
int g = GetParameter<int>("g");
root.ImageData[x,y].G = g;
int b = GetParameter<int>("b");
x = GetParameter<int>("x");
y = GetParameter<int>("y");
root.ImageData[x,y].B = b;
```

Конструктор процесса адаптации представления СД на языке программирования. Будем специализировать ОКПС для представления адаптера СД следующей структурой:

$$C = \langle M, \Sigma, \Lambda \rangle\ _S\mapsto C_{ADS} = \left\langle M_{ADS}, \Sigma_{ADS}, \Lambda_{ADS} \right\rangle,$$

где

$\Sigma_{ADS} =< \Xi_{ADS}, \Theta_{ADS}, \Phi_{ADS}, \{\rightarrow\} >$, $\Xi_{ADS} = \{".", "\cdot"\}$, $\Lambda_{ADS} = \Lambda \cup \Lambda_{20} \cup \Lambda_{21}$. $\Lambda_{ADS1} = \{M_{ADS} \supset (T_{ADS} \cup N_{ADS}),\ T_{ADS} = \{ B_i,\ i = 1...14 \},\ N_{SC} = \{ \alpha_i \} \}$.

Специализированный процесс представляет собой процесс структурной адаптации СД использующий критерий временной эффективности выполнения сценариев обработки данных и генетический алгоритм для случайного поиска оптимальной СД.

Частичная аксиоматика Λ_{20} приведена ниже.

Множество T_{ADS} состоит из приведенных ниже алгоритмов:

$B_1\big|_{\mathrm{P}}^{S_{\mathrm{P}}, R_{\mathrm{P}}}$ – получения S- и R- оценки показателей временной эффективности структур данных;

$B_2\big|_{\Omega(C_{PLi})}^{\Omega(C_{SC})}$ – генерации множества сценария;

$B_3\big|_{\Omega(C_{SC})}^{f_{SC}}$ – выбора сценария из множества сгенерированных;

$B_4\big|_{f_{SC}}^{RP}$ – формирования процесса выполнения RP из текста программной реализации и сценария;

$B_5\big|_{\Re}^{t}$ – получения текущего времени из таймера;

$B_6\big|_{RP,\Re}^{PE}$ – исполнения процесса выполнения сценария RP;

$B_7\big|_{a}^{b}$ – реализация операции присвоения без привязки к типу;

$B_8\big|_{t_{beg}, t_{end}, f_{PL}}^{\tilde{t}}$ – вычисления времени выполнения сценария программным шаблоном;

doi 10.15802/stp2016/67302

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)

Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2016, № 2 (62)

$B_9|^{b}_{\mathrm{P},S_\mathrm{P},R_\mathrm{P}}$ – вычисление признака завершения работы адаптера;

$B_{10}|^{\vec{r}}_{\Re,\vec{R}}$ – получение вектора случайных чисел на основе вектора $\vec{R}$, указывающего максимальные значения вектора $\vec{Q}$;

$\mathrm{B}_{11}|^{\mathrm{P}_j^-,r}_{\mathrm{P}_i,\mathrm{a}}$ – принятие решения относительно особи на основе оценки a, $true$, если особь должна остаться в популяции;

$B_{12}|^{a}_{\mathrm{P}}$ – подсчет оценки популяции;

$B_{13}|^{\upsilon_3}_{\upsilon_1,\upsilon_2}$ – скрещивание особи υ_1 и υ_2 в результирующую особь υ_3;

$B_{14}|^{R,f_{PL}}_{\Re,Q}$ – выполнение генерации конструкций текста преобразователя $\Omega(C_{PL})$.

$B_{15}|^{\Omega(\mathrm{C}_{LSD})}_{Or}$ – выполнение генерации конструкций логической структуры данных, на основе правил, построенных внешним исполнителем Or;

$B_{16}|^{\Omega(\mathrm{C}_{PLI})}_{f_{LSD}}$ – выполнение генерации конструкций $\Omega(C_{PLI})$ на основе конструкции f_{LSD};

$B_{17}|^{f_{LSD}}_{\Omega(\mathrm{C}_{LSD})}$ – выбор конструкции логической структуры из множества $\Omega(C_{LSD})$;

$B_{18}|^{k'}_{k}$ – инкрементация значения переменной.

В приведенных алгоритмах $\Omega(C_{PLI})$ – декларация классов на языке C#, формируемая как реализация преобразователя C_{PLI} каждой примитивной конструкции (с однородными и неоднородными элементами), декларируется свой класс с соответствующими методами обработки данных; $\Omega(C_{SC})$ – множество программных сценариев обработки данных; $\Omega(C_{PL})$ – множество текстов программ объектно-ориентированных структур данных, формируемых в результате реализации преобразователя C_{PL}, которые различаются вариантами физического представления в оперативной памяти логических структур данных (размещение данных в ОП и методы их обработки); $\vec{Q}$ – вектор с идентификаторами программных шаблонов, f_{PL} – программный шаблон, f_{SC} – сценарий обработки данных, RP – процесс выполнения сценария программным шаблоном, $\Re$ – программно-аппаратная среда адаптации структур данных, PE – выполнение сценария, $\vec{r}$ – вектор случайных чисел, P – популяция, t – время выполнения сценария программной реализацией, закодированной заданной особью, P′ – популяция с соответствующими значениями времени выполнения сценария, P^- – сокращённая популяция, P^* – популяция с размноженными особями, P_i,t_i – особь, время соотв. этой особи, b – признак продолжения работы адаптера.

Операции связывания терминалов и нетерминалов $x \cdot y$, где x и y – алгоритмы, которые выполняются последовательно.

Для определения интерпретации структуры C_{ADS} воспользуемся следующей базовой алгоритмической структурой:

$$C_{A,ADS} = \left\langle M_{A,ADS}, V_{A,ADS}, \Sigma_{A,ADS}, \Lambda_{A,ADS} \right\rangle,$$

где $M_{A,ADS}$ – неоднородный носитель, $V_{A,ADS}$ – множество образующих алгоритмов, $\Sigma_{A,ADS}$ – сигнатура и $\Lambda_{A,ADS}$ – аксиоматика. Носитель $M_{A,ADS} \supset T_{ADS} \cup N_{ADS} \cup \Omega(C_{A,ADS}) \cup W$, где $\Omega(C_{A,ADS})$ все сформированные процессы адаптации структур данных; W – множество допустимых значений атрибутов. Множество алгоритмов $V_{A,ADS} \supset \{ A_1^0|^{A_i \cdot A_j}_{A_i,A_j}$, $A_2^0|^{:A_i}_{Z_1,Z_2,A_i}$, $A_3|^{f_j}_{l_h,l_q,f_i}$, $A_4|^{f_j}_{f_i,\Psi}$, $A_5|^{f_j}_{f_i,\Psi}$, $\} \bigcup V_W \subset \Omega(C_{A,ADS})$.

$A_1^0|^{A_i \cdot A_j}_{A_i,A_j}$, $A_2^0|^{:A_i}_{Z_1,Z_2,A_i}$ такие как в $C_{A,LD}$, $A_3|^{f_j}_{l_h,l_q,f_i}$ такой как $A_5|^{f_j}_{l_h,l_q,f_i}$ в $C_{A,LD}$, $A_4|^{f_j}_{f_i,\Psi}$, такой как $A_6|^{f_j}_{f_i,\Psi}$ в $C_{A,LD}$, $A_5|^{f_j}_{f_i,\Psi}$ такой как $A_7|^{f_j}_{f_i,\Psi}$ в $C_{A,LD}$; $A_6|^{l_i,l_j}_{l_i,l_j}$ алгоритм связывания элементов, $l_i, l_j \in M_{SC}$;.

Интерпретация конструктора процесса адаптации:

$$< C_{ADS} =< M_{ADS}, \Sigma_{ADS}, \Lambda_{ADS} >,$$

$$C_{A,ADS} =< M_{A,ADS}, \Sigma_{A,ADS}, \Lambda_{A,ADS} >>$$

doi 10.15802/stp2016/67302

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)

Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2016, № 2 (62)

$$ {}_I\mapsto \ {}_I C_{ADS} =< M_{I,ADS}, \Sigma_{I,ADS}, \Lambda_{I,ADS} > $$

где ${}_I\mapsto$ – операция интерпретации; $\Lambda_{I,ADS} = \Lambda_{ADS} \cup \Lambda_{22}$; $\Lambda_{SC1} = \{ (A_1^0 \mid_{A_i, A_j}^{A_i \cdot A_j} \hookleftarrow \cdot);$ $(A_2^0 \mid_{Z_1, Z_2, A_i}^{A_i} \hookleftarrow :);$ $(A_3 \mid_{l_h, l_q, f_i}^{f_j} \hookleftarrow \Rightarrow);$ $(A_4 \mid_{f_i, \Psi}^{f_j} \hookleftarrow \mid\Rightarrow);$ $(A_5 \mid_{f_i, \Psi}^{f_j} \hookleftarrow \|\Rightarrow);$ $(A_6 \mid_{l_i, l_j}^{l_i, l_j}, \hookleftarrow \cdot) \}$

Рассмотрим конкретизацию для адаптера.

$$ {}_I C_{ADS} = \left\langle M_{ADS}, \Sigma_{ADS}, \Lambda_{I,ADS} \right\rangle $$

$$ {}_K\mapsto \ C_{K,ADS} = \left\langle M_{ADS}, \Sigma_{ADS}, \Lambda_{K,ADS} \right\rangle $$

где $M_{ADS} \supset T_{ADS} \cup N_{ADS}$; $\Sigma_{ADS} = \Sigma_{S,ADS}$; $\Lambda_{ADS} = \Lambda_{I,ADS} \cup \Lambda_{23} \cup \Lambda_{24}$.

$\Lambda_{22} =< T_{ADS} = \{T_{S,ADS}\}, \quad N_{ADS} = \{ \ \sigma, \ \nu, \ \delta, \ \mu \ \} >$.

Частичная аксиоматика Λ_{23} включает следующие конструктивные дополнения: *n* – колво первоначальных особей генетического алгоритма, задается внешним исполнителем; *a* – параметры генерации случайных чисел особей, задается внешним исполнителем на основании значения атрибута R $\Omega(C_{PL})$.

Частичная аксиоматика Λ_{24} состоит из следующих отношений подстановки.

Правило подстановки для подготовки процесса выполнения адаптации:

$$ \sigma \to B_{15} \mid_{Or}^{\Omega(C_{LSD})} \cdot B_{17} \mid_{\Omega(C_{LSD})}^{f_{LSD}} \cdot B_{16} \mid_{f_{LSD}}^{\Omega(C_{PLI})} \cdot $$

$$ B_2 \mid_{\Omega(C_{PLI})}^{\Omega(C_{SC})} \cdot B_3 \mid_{\Omega(C_{SC})}^{f_{SC}} \cdot \varphi \cdot \kappa \cdot \xi \, . $$

Правило для формирования вектора $\vec{R}$ для определения максимальных значений вектора $\vec{Q}$:

$$ \varphi \to B_{14} \mid_{\Re, \vec{Q}^0}^{\vec{R}, f_{PL}} . $$

Генерация первой популяции выполняется следующим правилом:

$$ \kappa \to \prod_{i=1}^{n} B_{10} \mid_{\Re, \vec{R}}^{\vec{Q}} \cdot B_{14} \mid_{\Re, \vec{Q}}^{\vec{R}, \mathrm{P}_i} . $$

Правило для определения времени выполнения сценария для каждой СД, зашифрованной особью популяции:

$$ \xi \to \prod_{i=1}^{n} \left(\begin{matrix} B_4 \mid_{f_{SC}, \mathrm{P}_i}^{RP} \cdot B_5 \mid_{\Re}^{t_{beg}} \times \\ \times B_6 \mid_{RP, \Re}^{PE} \cdot B_5 \mid_{\Re}^{t_{end}} \cdot B_8 \mid_{t_{beg}, t_{end}}^{t_i} \end{matrix} \right) \varsigma $$

Определение продолжения работы адаптера выполняется по следующему правилу:

$$ \varsigma \to B_1 \mid_{\mathrm{P}}^{S_\mathrm{P}, R_\mathrm{P}} \cdot B_9 \mid_{\mathrm{P}, S_\mathrm{P}, R_\mathrm{P}}^{b} \cdot A_2^0 \mid_{b, \{true\}}^{\nu \cdot \tau \cdot \xi} . $$

Правило для сокращения популяции на основе усредненной оценки времени выполнения сценария всей популяции. Здесь о – средняя оценка времени всей популяции:

$$ \nu \to B_{12} \mid_{\mathrm{P}}^{\mathrm{o}} \cdot B_7 \mid_0^k \cdot \prod_{i=1}^{n} (\mathrm{B}_{11} \mid_{\mathrm{P}_i, \mathrm{o}}^{\mathrm{P}_k^-, b} \cdot A_2^0 \mid_{t, \{true\}}^{B_{18} \mid_k^k}) \times $$

$$ \times B_7 \mid_{\mathrm{P}^-}^{\mathrm{P}} \cdot B_7 \mid_k^n $$

Генерация новых особей, на основе существующих в популяции, выполняется по следующему правилу:

$$ \tau \to B_7 \mid_0^k \cdot \prod_{i=1}^{n} \prod_{j=1}^{n} (B_{13} \mid_{\mathrm{P}_i, \mathrm{P}_j}^{\mathrm{P}^*_k} \cdot B_{18} \mid_k^k) \cdot B_7 \mid_{\mathrm{P}^*}^{\mathrm{P}} \cdot B_7 \mid_k^n ; $$

Реализация процесса адаптации выглядит следующим образом:

$$ B_{15} \mid_{Or}^{\Omega(C_{LSD})} \cdot B_{17} \mid_{\Omega(C_{LSD})}^{f_{LSD}} \cdot B_{16} \mid_{f_{LSD}}^{\Omega(C_{PLI})} \cdot B_2 \mid_{\Omega(C_{PLI})}^{\Omega(C_{SC})} $$

$$ \cdot B_3 \mid_{\Omega(C_{SC})}^{f_{SC}} \cdot B_{14} \mid_{\Re, \bar{Q}^0}^{\vec{R}, f_{PL}} \cdot \prod_{i=1}^{n} (B_{10} \mid_{\Re, \vec{R}}^{\bar{Q}} \cdot B_{14} \mid_{\Re, \bar{Q}}^{\vec{R}, \mathrm{P}_i}) $$

$$ \cdot \prod_{i=1}^{n} (B_4 \mid_{f_{SC}, \mathrm{P}_i}^{RP} \cdot B_5 \mid_{\Re}^{t_{beg}} \cdot B_6 \mid_{RP}^{PE} \cdot B_5 \mid_{\Re}^{t_{end}} \cdot B_8 \mid_{t_{beg}, t_{end}}^{t_i}) $$

$$ \cdot B_1 \mid_{\mathrm{P}}^{S_\mathrm{P}, R_\mathrm{P}} \cdot B_9 \mid_{\mathrm{P}, S_\mathrm{P}, R_\mathrm{P}}^{b} \cdot A_2^0 \mid_{b, \{true\}}^{B_{19}} \quad B_{19} = B_{12} \mid_{\mathrm{P}}^{\mathrm{o}} \cdot B_7 \mid_0^k \cdot $$

$$ \prod_{i=1}^{n} (\mathrm{B}_{11} \mid_{\mathrm{P}_i, \mathrm{o}}^{\mathrm{P}_k^-, b} \cdot A_2^0 \mid_{t, \{true\}}^{B_{18} \mid_k^k}) \cdot \quad B_7 \mid_{\mathrm{P}^-}^{\mathrm{P}} \cdot B_7 \mid_k^n \quad \cdot B_7 \mid_0^k \cdot $$

$$ \prod_{i=1}^{n} \prod_{j=1}^{n} (B_{13} \mid_{\mathrm{P}_i, \mathrm{P}_j}^{\mathrm{P}^*_k} \cdot B_{18} \mid_k^k) \cdot B_7 \mid_{\mathrm{P}^*}^{\mathrm{P}} \cdot B_7 \mid_k^n \cdot \quad \prod_{i=1}^{n} ($$

$$ B_4 \mid_{f_{SC}, \mathrm{P}_i}^{RP} \quad \cdot B_5 \mid_{\Re}^{t_{beg}} \cdot B_6 \mid_{RP, \Re}^{PE} \cdot B_5 \mid_{\Re}^{t_{end}} \cdot B_8 \mid_{t_{beg}, t_{end}}^{t_i}) $$

$$ \cdot B_1 \mid_{\mathrm{P}}^{S_\mathrm{P}, R_\mathrm{P}} \cdot B_9 \mid_{\mathrm{P}, S_\mathrm{P}, R_\mathrm{P}}^{b} \cdot A_2^0 \mid_{b, \{true\}}^{B_{19}} $$

Циклический процесс структурной адаптации представления СД включает:

− синтез СД с разной программной реализацией;

doi 10.15802/stp2016/67302

− выполнение сценария с использованием синтезированного программного кода;

− анализ временной эффективности и выбор лучшей программной реализации СД.

Синтез СД заключается в генерации программного текста библиотеки классов реализующей заданную логическую структуру данных, и ее компиляции в промежуточный или бинарный код.

Научная новизна и практическая значимость

Впервые предложена конструктивная модель процессов разработки и адаптации структур данных к различным программно-аппаратным средам. При этом адаптируется размещение данных в ОП и алгоритмы их обработки.

Применение конструктивизма в моделировании позволило в рамках единого подхода и применяемых средств связать модели данных и алгоритмы их обработки с критериями эффективности.

Усовершенствована методика формирования системы КПС, механизмы, связи между взаимодополняющими друг друга КПС. Модификация конструктора и преобразователей позволяет коренным образом изменять и исследовать процесс адаптации

Разработанная модель позволяет автоматизировать процессы рационального размещения данных в ОП, что, в свою очередь, повышает временную эффективность программ со значительной долей обработки больших и очень больших объемов данных.

Выводы

Формализация представления данных на логическом уровне, преобразование логического представления СД в текст программы, конструирование текста программы сценариев, процесс адаптации выполнены средствами конструкционно-продукционных структур.

Модификация конструктора и преобразователей позволяет коренным образом изменять процесс адаптации:

− формировать на логическом уровне необходимые СД;

− применять различные подходы и способы формирования;

− совершенствовать алгоритмы адаптации и применять различные критерии качества СД.

При данном подходе разработчику программных систем достаточно в редакторе Dia [11, 12] построить логическую структуру данных, задать параметры сценариев. Тогда реализация системы конструктивно-продукционных структур сформирует тексты методов доступа к данным на ЯП C# для адаптированного к определенной программно-аппаратной среде размещения данных в оперативной памяти.

В соответствии с современными парадигмами программирования [9] разработаны соответствующие инструментальные программные средства [11, 12].

Универсализм дает возможность применять предложенный подход для улучшения временных характеристик программных средств, предполагающих значительные временные затраты. Этот подход позволяет существенно улучшить временные характеристики за счет рационального размещения данных в ОП.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Акуловский, В. Г. Алгебра для описания данных в композиционных схемах алгоритмов / В. Г. Акуловский // Проблеми програмування. – 2012. – № 2-3. – С. 234–240.
2. Акуловский, В. Г. Основы алгебры алгоритмов, базирующейся на данных / В. Г. Акуловский // Проблеми програмування. – 2010. – № 2−3. – С. 89–96.
3. Алгеброалгоритмические модели и методы параллельного программирования / Ф. И. Андон, А. Е. Дорошенко, Г. Е. Цейтлин, Е. А. Яценко. – Киев : Академпериодика, 2007. – 634 с.
4. Глушков, В. М. Алгебра. Языки. Программирование / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко. – Київ : Наук. думка, 1978. – 319 с.
5. Данные в языках программирования: абстракция и типология / под ред. В. Агафонова. – Москва : Мир, 1982. – 328 с.
6. Дорошенко, А. Е. Алгебра алгоритмов с данными и прогнозирование вычислительного процесса / А. Е. Дорошенко, В. Г. Акуловский // Проблеми програмування. – 2011. – № 3. – С. 3–10.
7. Дрождин, В. В. Анализ эффективности и эволюция структур данных / В. В. Дрождин, В. М. Володин // Проблемы информатики в образовании, управлении, экономике и технике : сб. ст. IX

doi 10.15802/stp2016/67302

Междунар. науч.-техн. конф. / Приволж. Дом знаний. – Пенза, 2009. – С. 33–44.

8. Ефимушкина, Н. В. Исследование подсистемы «кэш-память-оперативная память» многопроцессорных вычислительных систем / Н. В. Ефимушкина, М. М. Ефремов, С. П. Орлов // Вестн. Самар. гос. техн. ун-та. Серия: Техн. науки. – Самара, 2012. – № 3 (35). – С. 49–55.
9. Лаврищева, Е. М. Software Engineering компьютерных систем. Парадигмы, технологии и CASE-средства программирования / Е. М. Лаврищева. – Київ : Наук. думка, 2013. – 283 с.
10. Шинкаренко, В. И. Конструктивная модель адаптации структур данных в оперативной памяти: Часть I. Конструирование текстов программ / В. И. Шинкаренко, Г. В. Забула // Наука та прогрес транспорту. – 2016. – № 1 (61). – С. 100–112. doi: 10.15802/stp2016/61003.
11. Шинкаренко, В. И. Повышение временной эффективности структур данных в оперативной памяти на основе адаптации / В. И. Шинкаренко, Г. В. Забула // Проблеми програмування. – 2012. – № 2-3. – С. 211–218.
12. Шинкаренко, В. И. Применение генетического алгоритма в задачах адаптации структур данных / В. И. Шинкаренко, Г. В. Забула // Искусственный интеллект. – 2012. – № 3. – С. 323–331.
13. Array Based HV/VH Tree: an Effective Data Structure for Layout Representation / J. Ren, W. Pan, Y. Zheng [et al.] // J. of Zhejiang University-SCIENCE C. – 2012. – Vol. 13. – Iss. 3. – P. 232–237. doi: 10.1631/jzus.c1100193.
14. Attali, D. Efficient Data Structure for Representing and Simplifying Simplicial Complexes in High Dimensions / D. Attali, A. Lieutier, D. Salinas // Intern. J. of Computational Geometry & Applications. – 2012. – Vol. 22. – Iss. 4. – P. 279–303. doi: 10.1142/S0218195912600060.
15. Bastani, F. B. The effect of data structures on the logical complexity of programs / F. B. Bastani, S. S. Iyengar // Communication of the ACM. – 1987. – Vol. 30. – Iss. 3. – P. 250–259. doi: 10.1145/214748.214760.
16. Shynkarenko, V. I. Constructive-Synthesizing Structures and Their Grammatical Interpretations. I. Generalized Formal Constructive-Synthesizing Structure / V. I. Shynkarenko, V. M. Ilman // Cybernetics and Systems Analysis. – 2014. – Vol. 50. – Iss. 5. – P. 655–662. doi: 10.1007/-s10559-014-9655-z.
17. Weiss, M. A. Data Structures and Algorithm Analysis in C++ / M. A. Weiss. – London : Pearson Education Inc., 2014. – 656 p.

В. І. ШИНКАРЕНКО[1*], Г. В. ЗАБУЛА[2*]

[1*]Каф. «Комп'ютерні інформаційні технології», Дніпропетровський національный університет залізничного транспорту імені академіка В. Лазаряна, вул. Лазаряна, 2, Дніпропетровськ, Україна, 49010, тел. +38 (056) 373 15 35, ел. пошта shinkrenko_vi@ua.fm, ORCID 0000-0001-8738-7225

[2*]Каф. «Комп'ютерні інформаційні технології», Дніпропетровський національный університет залізничного транспорту імені академіка В. Лазаряна, вул. Лазаряна, 2, Дніпропетровськ, Україна, 49010, тел. +38 (056) 373 15 35, ел. пошта zabulus12@gmail.com, ORCID 0000-0002-8607-5729

КОНСТРУКТИВНА МОДЕЛЬ АДАПТАЦІЇ СТРУКТУР ДАНИХ В ОПЕРАТИВНІЙ ПАМ'ЯТІ: ЧАСТИНА ІІ. КОНСТРУКТОРИ СЦЕНАРІЇВ І ПРОЦЕСІВ АДАПТАЦІЇ

Мета. Друга частина статті завершує представлення системи конструктивно-продукційних структур (КПС), що моделюють адаптацію структур даних в оперативній пам'яті (ОП). Мета другої частини дослідження полягає в розробці моделі процесу адаптації даних в ОП, що функціонують у різноманітних програмно-апаратних середовищах, та сценаріїв процесів обробки даних. **Методика.** Для впровадження мети застосована методологія математико-алгоритмічного конструктивізму. У даній частині роботи, на основі узагальненої КПС, шляхом її трансформаційних перетворень розробляються конструктори сценаріїв і процесів адаптації. Конструктори є інтерпретованими, спеціалізованими КПС. Виділяються термінальні алфавіти конструктора сценаріїв у вигляді алгоритмів обробки даних і конструктора адаптації – у вигляді алгоритмічних складових процесу адаптації. Методика передбачає розробку правил підстановки, що визначають процес виводу відповідних конструкцій. **Результати.** У другій частині статті представлена частина системи КПС, що моделює адаптацію розміщення даних в ОП, а саме, конструкторів сценаріїв та процесів адаптації. Результатом реалізації конструктора сценаріїв є набір операцій обробки даних у вигляді тексту мовою програмування C#, конструктора процесу адаптації – процес адаптації, а результатом процесу

doi 10.15802/stp2016/67302

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)

Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2016, № 2 (62)

адаптації – адаптований бінарний код обробки структур даних. **Наукова новизна.** Вперше запропонована конструктивна модель процесу обробки даних – сценарію, що враховує порядок та кількість звернень до різних елементів структур даних, а також адаптації структур даних до різноманітних програмно-апаратним середовищ. При цьому адаптується розміщення даних в ОП та алгоритми їх обробки. Застосування конструктивізму в моделюванні дозволило в рамках єдиного підходу і застосовуваних засобів зв'язати моделі даних та алгоритми їх обробки з критеріями ефективності. Розроблені моделі дозволяють досліджувати процес адаптації та керувати ним. **Практична значимість.** Розроблені модель та методи дозволяють автоматично змінювати розміщення даних в ОП та їх алгоритмічні зв'язки в залежності від експлуатаційних потреб, конструктивних особливостей апаратних засобів і програмного середовища функціонування.

Ключові слова: структура даних; конструктивно-продукційна структура; адаптація; конструктор; перетворювач

V. I. SHYNKARENKO[1*], H. V. ZABULA[2*]

[1*]Dep. «Computer and Information Technologies», Dnipropetrovsk National University of Railway Transport named after Academician V. Lazaryan, Lazaryan St., 2, Dnipropetrovsk, Ukraine, 49010, tel. +38 (056) 373 15 35, e-mail shinkrenko_vi@ua.fm, ORCID 0000-0001-8738-7225
[2*]Dep. «Computer and Information Technologies», Dnipropetrovsk National University of Railway Transport named after Academician V. Lazaryan, Lazaryan St., 2, Dnipropetrovsk, Ukraine, 49010, tel. +38 (056) 373 15 35, e-mail zabulus12@gmail.com, ORCID 0000-0002-8607-5729

CONSTRUCTIVE MODEL OF ADAPTATION OF DATA STRUCTURES IN RAM. PART II. CONSTRUCTORS OF SCENARIOS AND ADAPTATION PROCESSES

Purpose. The second part of the paper completes presentation of constructive and the productive structures (CPS), modeling adaptation of data structures in memory (RAM). The purpose of the second part in the research is to develop a model of process of adaptation data in a RAM functioning in different hardware and software environments and scenarios of data processing. **Methodology.** The methodology of mathematical and algorithmic constructionism was applied. In this part of the paper, changes were developed the constructors of scenarios and adaptation processes based on a generalized CPS through its transformational conversions. Constructors are interpreted, specialized CPS. Were highlighted the terminal alphabets of the constructor scenarios in the form of data processing algorithms and the constructor of adaptation – in the form of algorithmic components of the adaptation process. The methodology involves the development of substitution rules that determine the output process of the relevant structures. **Findings.** In the second part of the paper, system is represented by CPS modeling adaptation data placement in the RAM, namely, constructors of scenarios and of adaptation processes. The result of the implementation of constructor of scenarios is a set of data processing operations in the form of text in the language of programming C#, constructor of the adaptation processes – a process of adaptation, and the result the process of adaptation – the adapted binary code of processing data structures. **Originality.** For the first time proposed the constructive model of data processing – the scenario that takes into account the order and number of calls to the various elements of data structures and adaptation of data structures to the different hardware and software environments. At the same the placement of data in RAM and processing algorithms are adapted. Constructionism application in modeling allows to link data models and algorithms for their processing with the performance criteria in the framework of unified approach and applied means. The developed models allow us to study the process of adaptation and control it. **Practical value.** The developed model and methods allow automatically changing the data placement in the RAM and their algorithmic connection depending on the operational requirements, the design features of the hardware and software operating environment.

Keywords: data structure; constructive and productive structure; adaptation; designer; converter

REFERENCES

1. Akulovskiy V.G. Algebra dlya opisaniya dannykh v kompozitsionnykh skhemakh algoritmov [Algebra to describe the data in the compositional schemes of algorithms]. *Problemy prohramuvannia – Programming Problems*, 2012, no. 2-3, pp. 234-240.

doi 10.15802/stp2016/67302

2. Akulovskiy V.G. Osnovy algebry algoritmov, baziruyushcheysya na dannykh [Basic algebra algorithms based on data]. *Problemy prohramuvannia – Programming Problems*, 2010, no. 2-3, pp. 89-96.
3. Andon F.I., Doroshenko A.Ye., Tseytlin G.Ye., Yatsenko Ye.A. *Algebro-algoritmicheskiye modeli i metody parallelnogo programmirovaniya* [Algorithmic algebraic models and methods of parallel programming]. Kyiv, Akademperyodyka Publ., 2007. 634 p.
4. Glushkov V.M., Tseytlin G.Ye., Yushchenko Ye.L. *Algebra. Yazyki. Programmirovaniye* [Algebra. Languages. Programming]. Kyiv, Naukova dumka Publ., 1978. 319 p.
5. Agafonov V. *Dannyye v yazykakh programmirovaniya: abstraktsiya i tipologiya* [Data in programming languages: abstraction and typology]. Moscow, Mir Publ., 1982. 328 p.
6. Doroshenko A.Ye., Akulovskiy V.G. Algebra algoritmov s dannymi i prognozirovaniye vychislitelnogo protsessa [Algebra of algorithms with data and prediction of computational process]. *Problemy prohramuvannia – Programming Problems*, 2011, no. 3, pp. 3-10.
7. Drozhdin V.V., Volodin V.M. Analiz effektivnosti i evolyutsiya struktur dannykh [Analysis of the effectiveness and evolution of data structures]. *Sbornik statey IX Mezhdunarodnoy nauchno-tekhnicheskoy konferentsii «Problemy informatiki v obrazovanii, upravlenii, ekonomike i tekhnike»* [Proc. of IX Intern. Sci. and Techn. Conf. «Problems of Informatics in education, management, economics and technology»]. Penza, 2009, pp. 33-44.
8. Yefimushkina N.V., Yefremov M.M., Orlov S.P. Issledovaniye podsistemy «kesh-pamyat-operativnaya pamyat» mnogoprotsessornykh vychislitelnykh system [The study of subsystem «cache-memory-RAM» multiprocessor computational systems]. *Vestnik Samarskogo gosudarstvennogo tekhnicheskogo universiteta. Seriya: Tekhnicheskiye nauki* [Bulletin of Samara State Technical University. Series: Technical Science], 2012, no. 3 (35), pp. 49-55.
9. Lavrishcheva Ye.M. *Software Engineering kompyuternykh sistem. Paradigmy, tekhnologii i CASE-sredstva programmirovaniya* [Software Engineering of computer systems. Paradigms, technologies and CASE tools of programming]. Kyiv, Naukova dumka Publ., 2013. 283 p.
10. Shynkarenko V.I., Zabula H.V. Konstruktivnaya model adaptatsii struktur dannykh v operativnoy pamyati. Chast I. Konstruirovaniye tekstov programm [Constructive model of data structures adaptation in RAM. Part I. Program text constructing]. *Nauka ta prohres transportu – Science and Transport Progress*, 2016, no. 1 (61), pp. 100–112. doi: 10.15802/stp2016/61003.
11. Shinkarenko V.I., Zabula G.V. Povysheniye vremennoy effektivnosti struktur dannykh v operativnoy pamyati na osnove adaptatsii [Improving time efficiency of data structures in memory-based adaptation]. *Problemy prohramuvannia – Programming Problems*, 2012, no. 2-3, pp. 211-218.
12. Shinkarenko V.I., Zabula G.V. Primeneniye geneticheskogo algoritma v zadachakh adaptatsii struktur dannykh [Application of genetic algorithm in problems of adaptation of data structures]. *Iskusstvennyy intellect – Artificial Intelligence*, 2012, no. 3, pp. 323-331.
13. Ren J., Pan W., Zheng Y., Shi Z., Yan X. Array Based HV/VH Tree: an Effective Data Structure for Layout Representation. *Journal of Zhejiang University-SCIENCE C*, 2012, vol. 13, issue 3, pp. 232-237. doi: 10.1631/jzus.c1100193.
14. Attali D., Lieutier A., Salinas D. Efficient Data Structure for Representing and Simplifying Simplicial Complexes in High Dimensions. *Intern. Journal of Computational Geometry & Applications*, 2012, vol. 22, issue 4, pp. 279-303. doi: 10.1142/S0218195912600060.
15. Bastani F.B., Iyengar S.S. The effect of data structures on the logical complexity of programs. *Communication of the ACM*, 1987, vol. 30, issue 3, pp. 250-259. doi: 10.1145/214748.214760.
16. Shynkarenko V.I., Ilman V.M. Constructive-Synthesizing Structures and Their Grammatical Interpretations. I. Generalized Formal Constructive-Synthesizing Structure. *Cybernetics and Systems Analysis*, 2014, vol. 50, issue 5, pp. 655-662. doi: 10.1007/s10559-014-9655-z.
17. Weiss M.A. Data Structures and Algorithm Analysis in C++. London, Pearson Education Inc. Publ., 2014. 656 p.

Статья рекомендована к публикации д.т.н., проф. В. В. Скалозубом (Украина); д.физ.-мат.н., проф. В. Е. Белозёровым (Украина)

Поступила в редколлегию: 29.02.2016
Принята к печати: 17.04.2016

