УДК 004.896;004.94; 67.02

О.Д. Криськов, доц., канд. техн. наук, О.С. Шадурський, магістрант
Кіровоградський національний технічний університет

Модуль уточнення розрахункового значення по преференційному ряду в системах автоматизованого рішення конструкторських та технологічних задач

Розроблено алгоритм та в середовищі Delphi реалізовано відповідний модуль уточнення розрахункового значення згідно преференційного ряду. Наведено приклад його застосування при уточненні розрахункового значення числа обертів по паспортним даним верстата та один з можливих варіантів формування мікробази преференційних рядів.
системи автоматизованого проектування, технологічний процес, уточнення, преференційний ряд

Програмоване рішення інженерних розрахунків в сфері технологічної підготовки машинобудівного виробництва стикається з типовою задачею уточнення результату розрахунку по преференційному ряду чисел.

Так, виконуючи розрахунок валу на міцність, ми змушені діаметри його опорних шийок приймати відповідно стандартним діаметрам посадочних місць підшипників, а по іншим діаметрам орієнтуватись на преференційні ряди чисел згідно ГОСТ [1]. Аналогічна ситуація виникає при розрахунках модулів зубчатих коліс відповідно до заданого крутного моменту, колової швидкості обертання та матеріалу колеса. Адже значення модулів стандартизовані. В технологічних розрахунках типовою є задача уточнення розрахункового значення частоти обертання шпинделя чи величини подачі по паспортним даним верстата, які в даному разі можна розглядати як преференційні. При більш широкому підході до цієї проблеми виявляється, що аналогічна ситуація виникає і при алгоритмізації задачі призначення режимів різання по існуючим табличним нормативам [2,3]. Правда, тут з'являється необхідність обробляти масиви чисел з ведучими та прикінцевими нулями. Але це вимагає лише деякого корегування загального алгоритму уточнення розрахункового числа по преференційному ряду. На підставі вищевикладеного можна стверджувати, що задача уточнення розрахункового числа (ЗУРЧ) по преференційному ряду є типовою, а питання розробки алгоритму її програмованого рішення є актуальним.

Розглянемо коротко можливий алгоритм її рішення. Нехай на числовій вісі в певному масштабі відкладено значення чисел a_0, a_1, a_2... a_n , які розглядаються як преференційний ряд. Якщо в тому ж масштабі на цій вісі відкласти деяке число - результат певного розрахунку (РПР), то, в принципі, може виникнути шість варіантів співпадання РПР з елементами ряду (рис. 1).

Перший та шостий варіанти співпадання безумовно виходять за межі преференційного ряду, другий та п'ятий РПР також виходить за межі, але можуть бути

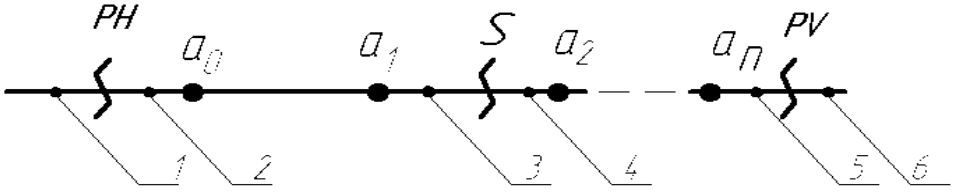

Рисунок. 1- Схема розмішення розрахункової величини на числовій вісі

прийнятними з певними застереженнями. В такому разі в якості преференційного числа умовимось прийняти мінімальне чи максимальне значення ряду. Нарешті, 3 та 4 варіанти співпадання, коли РПР розміщується між *i-тим* та *i+1* значеннями ряду. У таких випадках необхідно передбачити певний алгоритм прийняття рішення: при яких умовах приймати менше, а при яких - більше значення діапазону, в який попадає РПР.

Для того щоб алгоритм ЗУРЧ по преференційному ряду був універсальним введемо поняття нижнього (PH) та верхнього (PV) порогових значень ряду та рівня внутрішньої межі (S), які будемо визначати за виразами:

$$PH := a_0 \cdot (1 - iPH/100\%), \quad (1)$$

$$PV := a_n \cdot (1 + iPV/100\%), \quad (2)$$

$$S := a_i + (a_{i+1} - a_i) \cdot (100 - iLiB)/100\%. \quad (3)$$

Задаючи значення коефіцієнтів *iPH*, *iPV* та *iLib* у відсотках (від нуля до 100) ми будемо мати змогу впливати на конкретні числові значення указаних меж. Величини згаданих коефіцієнтів повинні вибиратись в залежності від технічної суті задачі. В процесі відладки та подальшої експлуатації модуля важливим є питання стійкості процедур при їх виклику, можливо навіть з некоректними з точки зору технічної суті даними. Таке звертання може мати місце, наприклад, внаслідок некоректного відпрацювання попередньої процедури в потоці, що реалізує розрахунковий алгоритм, як результат помилки в попередній процедурі і т.д. Незважаючи на те, що помилка, яка виникла, буде передаватись по ланцюжку від процедури до процедури, бажано, не переривати обчислення зразу ж на місці її появи, а продовжувати їх згідно алгоритму наскільки це буде можливо. Тоді за один прогін процедур модуля, по-перше, пройдемо далі по алгоритму основної вирішуючої процедури і тим самим будемо мати змогу виявити і інші, додаткові помилки, що можуть мати місце, а по-друге, будемо мати змогу упевнитись в якості та надійності послідуючих процедур при непередбачених результатах роботи процедур попередніх. Нерідко це веде до уточнення алгоритму підпорядкованих процедур. Виходячи з вищесказаного, алгоритмом процедури необхідно передбачити, щоб у першому та шостому варіанту співпадання за преференційні приймались відповідно мінімальне та максимальне значення ряду, але при цьому генерувалось відповідне ситуаційне повідомлення. Так, наприклад, при мінімальному значені ряду рівному 10, значенні мінімальної межі рівним 9,5 (при iPH=5%) і розрахунковому значенні 9,4 приймемо преференційне значення 10, одночасно згенерувавши повідомлення про те, що мав місце перший (назвемо його некоректним) варіант співпадання РПР та елементів ряду. Користувач може визначити, чи приймати значення 10, як прийнятне, чи ні. В останньому випадку його необхідно програмно обробити. Так, якщо розрахункове значення буде, наприклад 9,7, тобто попадає між мінімальним значенням ряду і мінімальним пороговим значенням, то домовимось прийняти як преференційне мінімальне значення ряду без будь-яких застережень. Симетричну ситуацію будемо мати, якщо РПР буде більшим максимального порогового, або буде попадати між максимальним значенням останнього елементу ряду та максимальним пороговим. При цьому мається на увазі, що ряд чисел, який трактується як преференційний, упорядкований по висхідній, тобто кожний наступний елемент більший від попереднього.

Якщо порівняння РПР з числами ряду відповідає третьому чи четвертому варіантам, коли він попадає в проміжок між двома значеннями ряду, то важливим питанням є положення внутрішньої межі, яке визначається коефіцієнтом iLiB. Приймаючи iLiB=50%, згідно формули (3) ми приймаємо положення внутрішньої межі рівно на середині інтервалу. В такому разі логічно прийняти за преференційне менше число діапазону, при умові що РПР менше значення внутрішньої межі, і більше - в

протилежному випадку. Якщо проаналізувати формулу (3), то можна зауважити, що при iLiB = 100% вихідним буде завжди мінімальне, а при iLiB = 0% - максимальне значення інтервалу. Викладений вище алгоритм було реалізовано у вигляді процедури, блок-схема якої приведена на рис. 2.

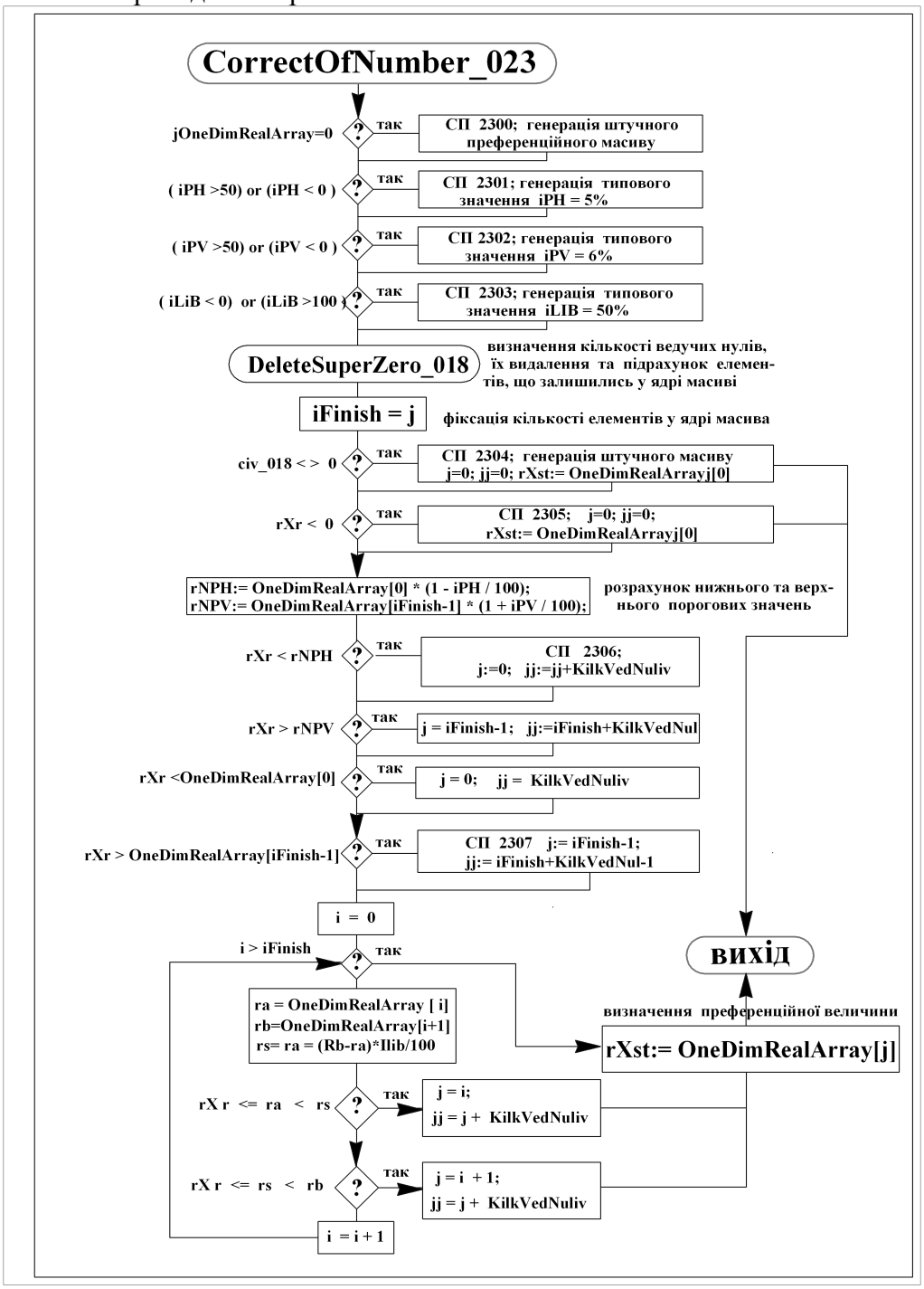

Рисунок 2 - Блок-схема процедури уточнення розрахункової величини

Останнім часом прогнозується тенденція [4] переходу від випуску закритих пакетних (фірмових) рішень, орієнтованих на певні платформи, до їх поділу на більш дрібні елементи, наприклад, окремі модулі, що вирішують виокремленні, логічно завершені частини певної інженерної задачі, наприклад, призначення режиму обробки при проектуванні технологічної операції, рішення ЗУРЧ і т.п. Такий підхід до розробки

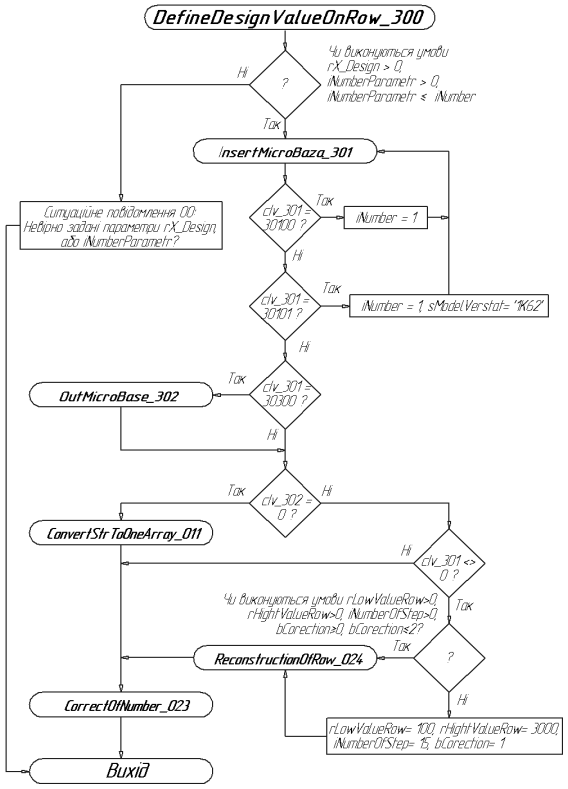

Рисунок 3 - Основана вирішуюча процедура

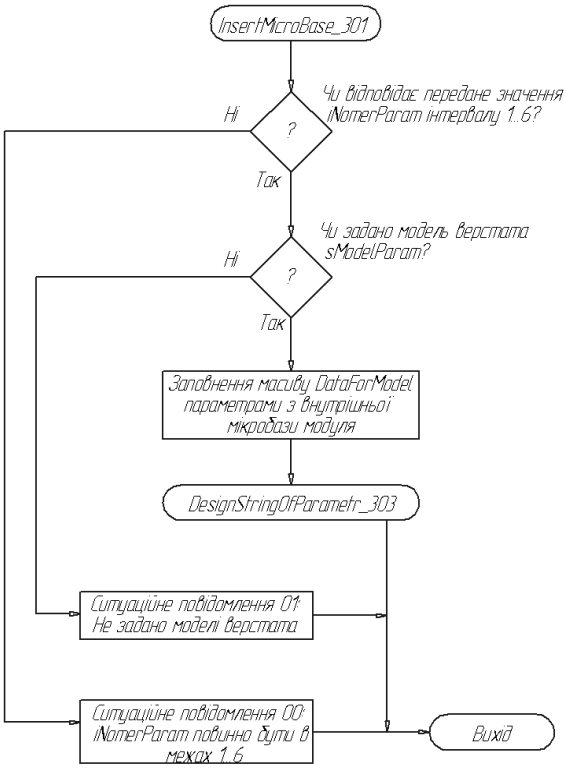

Рисунок 4 - Процедура – внутрішня база даних

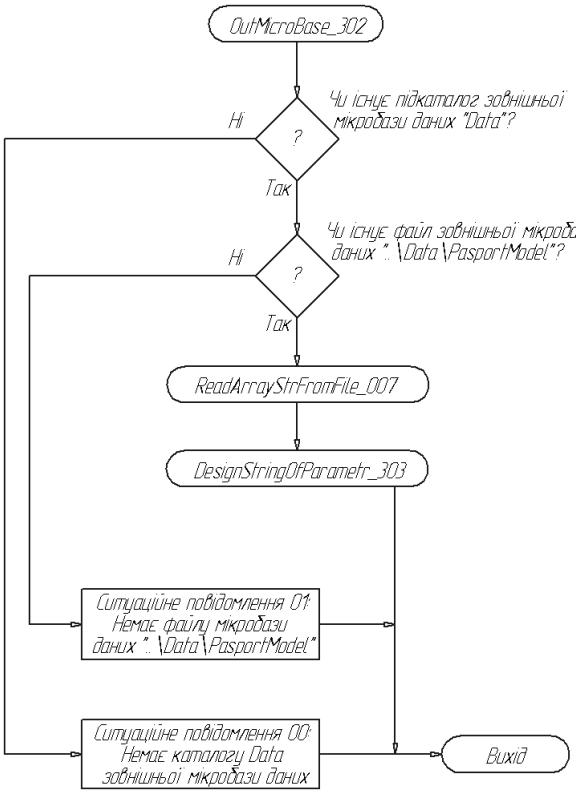

Рисунок 5 - Процедура – зовнішня база даних

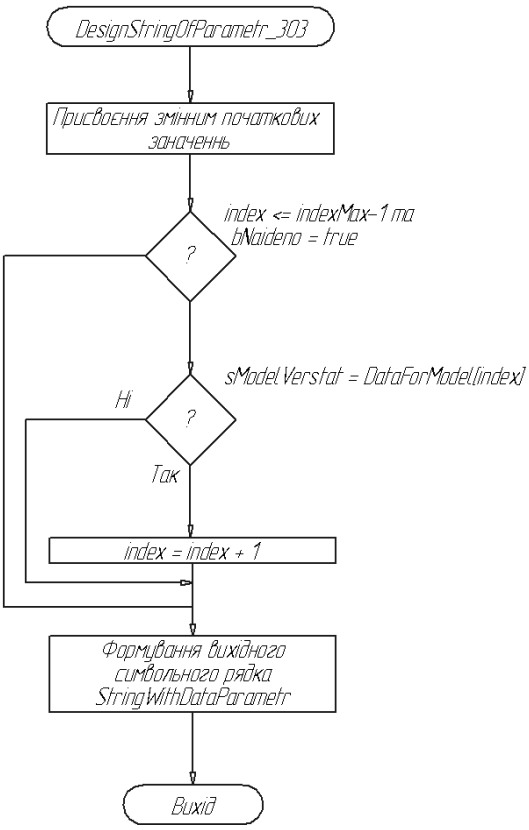

Рисунок 6 - Процедура вибору рядка даних

САПР дозволяє користувачам оперативно та гнучко реагувати на зміну ринкової ситуації.

В зв'язку з викладеним автори вважали доцільним розробити автономний універсальний алгоритм та відповідний динамічний модуль рішення ЗУРЧ в складі процедур DefineDesignValueOnRow_300, InsertMicroBaza_301, OutMicroBase_302, DesignStringOfParametr_303, кожна з яких має вузькоспеціальне призначення. Блок схеми цих процедур приведені на рис. 3 - 6. Процедура _300 в пакеті є основна вирішуюча. В залежності від змісту вхідних даних вона викликає або _301 внутрішню, або _302 зовнішню базу даних. Останні у відповідності з заданими вхідними параметрами вибирають у символьному виді преференційний ряд чисел з вищезгаданих баз. Вибраний рядок символів _303 процедура конвертує в одномірний динамічний реальний упорядкований по висхідній масив, який по суті є преференційним рядом і передає його _300-й, а далі разом з РПР - _023-й процедурі. Остання приймає рішення відповідно раніше описаному алгоритму. Як видно з блок-схем алгоритмів для рішення поставленої задачі використовується ряд процедур з пакету базових процедур (ПБП) _dll_technol_xxxxxx [5]. Процедура, названа зовнішньою базою, обробляє текстові файли, які, з додержанням певних правил, можуть бути підготовлені для будь-якої групи даних: діаметрів підшипників, модулів зубчатих коліс, ряду чисел обертів чи подач верстата і таке інше. Таким чином, з метою підвищення гнучкості системи проект має механізми доповнення бази даних новими рядами. Тим самим одночасно вирішено питання практично необмеженого розширення бази даних для преференційних рядів.

Тому враховуючи вищезгадане, _300 процедура включає чотири можливі варіанти формування преференційного ряду:

– якщо заданий ряд є у внутрішній мікробазі даних модуля, то уточнення виконується відповідно до нього;

– якщо заданий ряд відсутній в мікробазі даних модуля, але є в текстовому файлі PasportModel.txt, то уточнення проводиться згідно з ним;

– якщо заданий ряд відсутній і в мікробазі модуля і в текстовому файлі, але задані всі чотири параметри rLowValueRow, rHightValueRow, iNumberOfStep та bCorection для відпрацювання _024 процедури, то преференційний ряд, по якому проводиться уточнення, формується згідно з цими параметрами;

– якщо заданий ряд відсутній і в мікробазі модуля і в текстовому файлі, а хоча б один з вищезгаданих параметрів не задано, то _024 процедурою формується штучний рядок зі значеннями rLowValueRow = 100, rHightValueRow = 3000, iNumberOfStep = 15 та bCorection = 0, прийнятими по замовчуванню.

Розглянемо алгоритм роботи модуля на прикладі задачі уточнення обертів верстата по розрахунковому значенню, яке обчислюється на основі нормативної швидкості різання та діаметра оброблюваної поверхні.

Приймемо, що як внутрішня так і зовнішня мікробази даних являє собою декілька груп по шість символьних рядків з паспортними даними верстатів, зібраних в один масив (внутрішня) або в один файл (зовнішня). В першому рядку кожної групи її ознака – назва моделі верстата. Наступні рядки – це числа обертів, поздовжніх та поперечних подач і таке інше. Як видно з блок-схеми рис. 3, основною вирішуючою в модулі є _300 процедура, яка викликається головною програмою проекту. Головна програма – це своєрідний місток між середовищем, в якому створено проект, та його основною вирішуючою програмою. На початку роботи кожної з процедур перевіряється коректність даних, що передаються, і якщо хоч якийсь параметр задано невірно, генерується так зване ситуаційне повідомлення. Далі викликається _301 процедура (внутрішня база даних). По заданим ознакам у ній шукається група

символьних рядків, які відповідають заданій моделі верстата. Якщо відповідна група знайдена, то вона передається _303 процедурі. В противному разі _300 процедура викликає _302 (зовнішню мікробазу даних), в якій продовжується пошук такої ж групи рядків. При цьому контролюється наявність файлу зовнішньої мікробази, який повинен знаходитися у підкаталозі проекту, наприклад під назвою Data. У разі відсутності такого підкаталогу чи відсутності в ньому необхідного файлу генерується відповідне ситуаційне повідомлення. Якщо файл знайдено, то _007 процедура, з уже згаданого ПБП процедур, виконує його зчитування і знову робиться спроба вибрати із змісту зчитаного групу символьних рядків, яка характеризує задану модель верстата. Якщо таку групу знайдено, то вона знову таки передається в _303 процедуру. Остання вибирає з групи рядків саме той, який може бути використаний як преференційний у відповідності до технічної суті вирішуємої задачі. Далі відбувається уточнення по даному ряду з використанням _023 процедури з ПБП.

У випадку, коли потрібну групу преференційних рядків не знайдено ні у внутрішній ні у зовнішній мікробазах, щоб не допустити втрату темпу і максимально продвинутись по алгоритму _300 процедури, _024 процедурою (з ПБП) автоматично генерується штучний реальний одномірний упорядкований по висхідній масив чисел, який в подальшому використовується як преференційний. Така автогенерація може відбуватися по двом гілкам алгоритму: або по заданим користувачем мінімальному та максимальному значенням, кількості елементів і закону (геометрична чи арифметична прогресії) ряду, або по значенням, передбаченим в _300 процедурі за замовчуванням. Останні дві гілки алгоритму до певної міри організаційно спрощують використання модуля в навчальній роботі технічних вузів при вивченні курсу САПР ТП.

Таким чином, у відповідності до світових тенденцій проектування сучасних САПР, в статті виокремлено і поставлено логічно завершену самостійну задачу уточнення по преференційному ряду результату певного, відповідного технічній суті задачі розрахунку, розроблено алгоритм самостійного модуля для її рішення в складі основної вирішуючої та групи підпорядкованих процедур, які забезпечують формування та поповнення бази даних. Алгоритм закодовано в середовищі Delphi 5 і відлагоджено до робочого стану.

Список літератури

1. ГОСТ 6636-69. Основные нормы взаимозаменяемости. Нормальные линиейные размеры.
2. Общемашиностроительные нормативы режимов резания и вспомогательного времени на работы, выполняемые на металлорежущих станках. Массовое производство. Изд. Третье.- М.: Машиностроение, 1974, - 136 с.
3. Общемашиностроительные нормативы времени и режимов резания на токарно-автоматные работы. Массовое, крупносерийное и серийное производство. – М.: Машгиз, 1962, - 283 с.
4. Журнал «СНІР. Компьютеры и коммуникации» №12 2004, «Будущее – за модульностью» с. 65
5. Автоматизація призначення режимів різання та технічного нормування при проектуванні технологічних процесів виготовлення деталей на металорізальних верстатах. Методичні вказівки №1046 для виконання лабораторних та курсових робіт та проектів. Укладач Криськов О.Д., Кіровоград, РВЛ КНТУ, -155 с.

Разработан алгоритм и в среде Delphi реализован соответствующий модуль уточнения расчетного значения согласно предпочтительного ряда. Приведён пример его использования для уточнения расчетного значения числа оборотов по паспортним даним станка и один из возможных вариантов формирования микробазы предпочтительных рядов.

Designed algorithm and in the ambience Delphi is work out respective module of revision of calculation value according to preferred row. Cite An Instance algorithm of its use for revision of calculation value of number of turns on passport is given tool and one of the possible variants of shaping a microbase preferred row.