ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ГЗ – глобальна змінна

ЛЗ – локальна змінна

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

СЧ – система числення.

ВСТУП

Планування матеріальних ресурсів та високоточні інженерні розрахунки є важливою складовою роботи будь-якого виробничого підприємства. Адже від того, наскільки оптимально та швидко будуть здійснені розрахунки матеріалів, буде залежати якість, ціна і конкурентоспроможність кінцевого виробу, рівень задоволеності замовників, а також перспектива розвитку підприємства. І навпаки, неспроможність швидко та точно розрахувати кількість необхідних матеріалів і розміри деталей веде до повного краху виробництва необхідних пристроїв, збільшення грошових, а разом з тим матеріальних та часових витрат, збитків виробника тощо.

На багатьох державних металообробних підприємствах заходи, що спрямовані на оптимізацію, збільшення швидкості та зручності інженерних розрахунків мають нерегулярний характер, та здебільшого є результатом особистих ініціатив. Причиною тому є державна форма власності, низька конкуренція між державними підприємствами, недостатні фінансові витрати на сучасну обчислювальну техніку, а також майже повна відсутність з боку управління підприємствами чітких вимог щодо показників ефективності виробничих процесів та задоволеності споживачів. До проблем сучасного вітчизняного машинобудування відносяться, зокрема, недостатня швидкість виконання інженерних розрахунків, велика середня тривалість пошуку необхідних довідкових даних і виконання високоточних інженерних розрахунків, низька зручність деякого програмного забезпечення та майже повна відсутність автоматизації обчислень із використанням довідкових даних.

Зовсім іншим чином йдуть справи у приватних менших, за розмірами підприємствах, – комерційна форма діяльності спонукає їх більш швидко та точно розраховувати кількість необхідних матеріалів та розміри деталей, а конкуренція на ринку виробництва різноманітних виробів змушує відслідковувати надмірні витрати часу та матеріальних ресурсів. Для середніх і великих підприємств наявність програмно-апаратного комплексу підтримки інженерних розрахунків, є важливим чинником комерційного успіху.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

У даний момент на більшості українських металообробних підприємствах інженерні розрахунки виконуються за допомогою електронної обчислювальної техніки та документації у паперовій і електронній формах. Розвинені активні підприємства здійснюють більшість обчислень та креслень за допомогою програм автоматизованого проектування таких як Mathcad, AutoCAD, Autodesk Inventor. Проте дуже часто виникають задачі, які не зручно розв'язувати за допомогою спеціалізованих програм. Офісні пакети, такі як Microsoft Office та LibreOffice, частково вирішують дану проблему, але не надають достатньої зручності, мобільності та інтерактивності. У таких випадках розвинуті підприємства звертаються в аутсорсингові компанії або власними силами розроблюють необхідне програмне забезпечення. На жаль, на деяких нестабільно працюючих заводах ситуація з автоматизацією інженерних розрахунків дещо гірша.

1.1 Огляд існуючих програмно-апаратних засобів для вирішення різноманітних задач у машинобудуванні

На вітчизняних державних машинобудівних підприємствах історично склалося так, що майже всі інженерні розрахунки виконуються так званим “гібридним” способом, тобто із використанням паперової та електронної документації, обчислювальної техніки. А мобільність, зручність та швидкість обчислень і пошуку необхідної довідкової інформації при цьому нехтується. Крім того, час, виділений працівнику на такі розрахунки, не піддавався ніякому плануванню, міг марно витрачатись на пошуки довідкової інформації та збільшення точності обчислень. Такий спосіб організації знижує видатки підприємства на розрахунки, але збільшує неекономічні видатки на матеріали та робочий час.

Певним удосконаленням засобів розрахунків, проектування і планування ресурсів стало використання такого програмного забезпечення як:

1С:Підприємство 8, Autodesk Inventor 2014, Microsoft Office 2003.

Система програм "1С:Підприємство" включає в себе платформу і прикладні рішення, розроблені для автоматизації діяльності організацій і приватних осіб. Сама платформа не є програмним продуктом для використання кінцевими користувачами, які зазвичай працюють з одним із багатьох прикладних рішень (конфігурацій), розроблених за допомогою даного ПЗ. Такий підхід дозволяє автоматизувати різні види діяльності, у тому числі ІР, використовуючи єдину технологічну платформу.

Платформа 1С:Підприємство 8 була створена з урахуванням досвіду застосування системи програм 1С:Підприємство 7.7. З 2003 року вона показує активний розвиток. Основне завдання платформи: надання середи розробки (framework) програмістам прикладних рішень. Платформа забезпечує розробникам підвищення рівня абстракції при створенні та використанні прикладних рішень. Це дозволяє перейти від технічних і низькорівневих понять до більш змістовних і високорівневих, наблизити ці поняття до мови користувачів і фахівців в предметної області. У кінцевому підсумку це значно прискорює і уніфікує розробку прикладного рішення та його підтримку. Одночасно з цим платформа вирішує й традиційні завдання, пов'язані з продуктивністю, ергономікою, функціональністю тощо.

На машинобудівних підприємствах даний програмний комплекс використовується в таких областях:

- автоматизація виробничих і торгівельних процесів, бюджетних і фінансових відділів;
- підтримка оперативного управління підприємством;
- автоматизація організаційної та господарської діяльності;
- ведення бухгалтерського обліку з декількома планами рахунків довільних розмірів, регламентація звітності;
- широкі можливості для управлінського обліку і побудови аналітичної документації, підтримка багатовалютного обліку;

– вирішення задач планування, бюджетування і фінансового аналізу;
– розрахунок зарплати і управління персоналом;

Сучасне життя вимагає від машинобудівних підприємств великої конкурентоспроможності. Не є виключенням і підприємства, що проектують та виробляють судові машини і механізми. Проектна документація повинна розроблюватись швидко та відповідати всім нормам, а об'єкт проектування – по можливості мати не тільки двовимірне зображення, але й, для більшої наочності, бути представленим у тривимірному вигляді. Отже, в сучасному виробництві досить широко використовуються різноманітні системи автоматизованого проектування (САПР).

Однією з таких систем є Autodesk Inventor – параметрична система тривимірного твердотільного та поверхневого проектування, що призначена для створення цифрових прототипів промислових виробів і їх проектної документації. Також у середовищі Autodesk Inventor можна виконувати інженерний аналіз виробу та перевірку його в дії. Новітній принцип проектування – це функціональне проектування, що не обмежується лише геометричним моделюванням.

Інструментальні засоби Autodesk Inventor забезпечують повний цикл конструювання та формування конструкторської документації. Дана програма допомагає машинобудівним підприємствам у таких основних напрямках:

– значне скорочення циклу розробки моделі конструкції;
– можливість спільної роботи над конструкцією всіх розробників, зокрема груп інженерів, які знаходяться на великій відстані один від одного;
– можливість вводу користувальних примітивів у параметричному вигляді з метою повторного використовування;
– забезпечення доступу до тримірної моделі конструкції не тільки розробникам, а й усім групам користувачів, які беруть участь в роботі над проектом.

Отже, в машинобудуванні використовуються такі функціональні

можливості Autodesk Inventor:

1. адаптивне конструювання (забезпечуються дуже високі часові й ергономічні характеристики деталей, а всі конструктивні елементи, які можуть бути використані в процесі роботи над проектом, завжди несуть в собі найостаннішу інформацію про всі операції та оновлення, що відбувалися з ними);
2. адаптивне компонування (при завершенні розробки загальної конструкції моделі система дозволяє змінити форму деталі, що представлена у вигляді рисунка чи ескізу, та накласти всі взаємні зв'язки для даної деталі в конструкції таким чином, як це було передбачено розробником на початковому етапі розробки конструкції);
3. вбудований конструктор елементів;
4. інструментарій, який забезпечує спільну роботу над документацією;
5. системи підтримки та супроводження процесу конструювання.

Сьогодні жодне підприємство не може уявити свою роботу без офісного пакету ПЗ – набору програм, призначених для роботи з електронною документацією. Склад офісного пакету, зазвичай, залежить від виробника та його спеціалізації, проте еволюція програмного забезпечення зробила типовими для офісних пакетів такі основні компоненти:

- текстовий процесор – для роботи з документів, що складаються з текстів, ілюстрацій, таблиць, графіків, даіграм тощо;
- редактор електронних таблиць – засіб обробки таблиць з даними;
- засіб створення презентацій – дозволяє створювати яскраві кольорові електронні презентації, щоб краще описати клієнтам пропозиції та роботу компанії;
- система управління базами даних в офісних пакетах, як правило, забезпечує початковий рівень функціональних можливостей створення, редагування та доповнення БД;
- графічний редактор – дозволяє редагувати зображення різних форматів.

Своє походження офісні пакети ведуть від програм обробки текстів. Перші повноцінні офісні пакети, що містять текстовий процесор, редактор електронних таблиць і різні утиліти, з'явились лише у 80-х роках. Піонером у сфері офісних пакетів вважається компанія Lotus, яка в 1982 році випустила першу версію табличного редактора Lotus 1-2-3 (займав усього 256 КБ ОЗП). В тому ж році компанія Microsoft випустила першу версію текстового редактора Word для DOS, через деякий час до якого були додані програми для проведення презентацій, планувальники-органайзери, а також персональні системи управління базами даних.

Основний етап еволюції офісних пакетів здійснювався з початку 80-х років до середини 90-х. Подальші удосконалення стосувалися лише покращання користувацького інтерфейсу, зручності у роботі, інтеграції в Інтернет та ін. Сучасні офісні продукти пропонують досить багато різних засобів, які часто не використовуються середньостатистичним користувачем.

Оскільки сучасні інформаційні технології впевнено рухаються в бік розвитку і втілення концепції «центральної цифрової нервової системи» підприємств, яка забезпечує доступ до будь-яких інструментів і даних, взаємозв'язок співробітників, стандартизацію і керування документообігом, то офісні пакети перетворились на офісні системи.

На більшості державних підприємств використовуються офісні продукти фірми Microsoft.

Microsoft Office являє собою інтегрований програмний пакет автоматизації офісної діяльності. Перші версії офісних програм Microsoft з’явилися в середині 80-х років. На той час вони працювали під управлінням операційної системи MS DOS. Перша Windows-версія пакету з’явилася з виходом Windows 3.0. Починаючи з 1995 року всі версії Microsoft Office є 32-бітними додатками, а сучасні включають у себе і 64-бітні реакції. З даного офісного пакету версії 2003 у машинобудуванні найчастіше використовуються такі програми:

– Microsoft Office Word – головний компонент офісного пакету і безперечно один із найпопулярніших текстових процесорів. Він містить дуже широкий

вибір інструментів для редагування та форматування текстів документів, а також для додання в них різноманітних об'єктів – математичних формул, растрових та векторних малюнків, графіків, діаграм і т. д. У Word 2003 включено нові функції, що спрощують створення, читання та спільне використання документів. У дану версію також включено підтримку формату XML як власного формату файлів.

- Microsoft Office ExceI – це потужний і простий у використанні процесор для обробки електронних таблиць. У даній версії програми збережено кращі традиції попередніх версій електронних таблиць. ExceI добре працює з діаграмами і графіками та може використовувати електронні таблиці в інших поширених форматах. В ExceI 2003 включено нові функції, що спрощують аналіз та спільне використання інформації.
- Microsoft Office Access – система керування базами даних з широкими можливостями обробки реляційних баз даних та створення нових складних програмних елементів. Вона дозволяє створювати БД, що можуть містити в собі робочі таблиці, запити, модулі, макроси, форми, звіти та web-сторінки. Зі створеними в Access БД дуже легко працювати навіть недосвідченим користувачам. Робота в такому ПЗ спрощується завдяки виявленню помилок, які часто зустрічаються, шляхом їх відмічання та представленням варіантів їх виправлення.
- Microsoft Office Power Point – це спеціалізована програма для створення комп'ютерних презентацій, які використовуються на ділових зустрічах, засіданнях фірм з навчальною метою і т. п. Вона дозволяє створювати набір слайдів, які автоматично чи в ручному режимі переглядаються на ПК, чи за допомогою цифрового проектора або розміщаються в Інтернеті. Слайди можна також роздрукувати на папері чи плівці. Файли Power Point 2003 можна легко записувати на компакт-диски, а завдяки інтеграції з плеєром Microsoft Windows Media Медіа при демонстрації слайдів можна відтворювати потокове аудіо та відео. Дана програма проста в роботі і має велику кількість інструментів, що дозволяє легко підготувати презентацію,

яка в доступній формі може виразити все, що користувач хоче повідомити.

Як було зазначено вище, усі розрахунки на більшості сучасних українських металообробних підприємствах виконуються лише за допомогою або звичайного кишенькового калькулятора, або аналогічних програм на мобільному телефоні (чи смартфоні) і ПК.

Калькулятор – це електронний обчислювальний пристрій для виконання операцій над числами або алгебраїчними формулами. Він замінив механічні обчислювальні пристрої такі як абаки, рахівниці, логарифмічні лінійки, механічні або електромеханічні арифмометри, а також математичні таблиці (перш за все - таблиці логарифмів).

У залежності від можливостей та цільової сфери використання калькулятори бувають найпростішими, бухгалтерськими, інженерними (науковими), фінансовими. До окремих класів, звичайно, виділяють програмовані калькулятори, які дають можливість виконання складних обчислень за попередньо завантаженою програмою. Спеціалізовані калькулятори призначені для виконання обчислень у дуже вузьких сферах (статистичні, медичні, спеціальні фінансові розрахунки і т. п.). Подібні калькулятори часто використовуються на підприємствах, але все частіше у вигляді програм для ПК.

Найпопулярнішою програмою такого класу серед машинобудівників є стандартний калькулятор Windows, який входить до складу операційної системи.

Програма Калькулятор повністю імітує роботу звичайного “апаратного” калькулятора, його кнопки можна натискати мишкою, також можливий ввід за допомогою цифрової клавіатури, вставка математичних виразів із буфера обміну. Вона дозволяє працювати в одній з двох форм: звичайній (вид «Обычный») або інженерному (вид «Инженерный»), тобто, фактично, викликати один з двох калькуляторів. Для зміни виду калькулятора потрібно в головному меню «Вид» вибрати відповідний пункт.

Звичайний калькулятор дозволяє виконувати 4 арифметичні дії (додавання, віднімання, множення, ділення), добувати квадратний корінь, знаходити число, обернене до даного, виконувати деякі дії з процентами, а також запам’ятовувати

одне число, – проміжний результат обчислень.

У інженерному режимі, який є найбільш вживаним робітниками підприємств, доступні такі додаткові функції:

- тригонометричні та гіперболічні функції, натуральний та десятковий логарифми, піднесення до степеня (для квадратів і кубів виділені окремі кнопки), добування кореня;
- перевід частин градуса в хвилини та секунди, обчислення факторіалів (для нецілого аргументу замість факторіала обчислюється гамма-функція Г(x+1));
- групування операцій (кнопки з дужками, є індикатор рівня вкладеності), переключення режимів відображення (фіксована або плаваюча точка), обчислення залишку від ділення.

1.2 Рішення виробничих задач

Більшість виробничих задач розв’язуються на підприємствах за допомогою довідкових даних. Усі вони необхідні для інженерних розрахунків і знаходяться в нормативних документах ГОСТ і ДСТУ. На підприємствах використовуються як паперові, так і електронні версії цих документів. Спробуємо висвітлити сутність кожного з видів цих актів.

Державний стандарт (рос. Государственный стандарт, ГОСТ; можливий український переклад абревіатури: ДЕСТ) – одна з основних категорій стандартів у СРСР, сьогодні міждержавного стандарту в СНД. Приймається Міждержавною радою зі стандартизації, метрології і сертифікації (МГС).

У радянські часи всі ГОСТи були обов'язковими для застосування в тих областях, які визначалися преамбулою самого стандарту.

Багато з українських національних стандартів ДСТУ, що існують нині, на сировину, продукцію, а також на методи виробництва виникли ще в радянські часи. Нові ГОСТи, головним чином загального характеру, з одного боку, ставлять галузь в жорсткі рамки – вони вказують з якої сировини і за якою технологією виробляти ті або інші продукти, а з іншого боку, дозволяють виробникам

зберігати свою індивідуальність.

До ухвалення відповідних технічних регламентів закон передбачає обов'язкове виконання вимог стандартів у частині, що стосується цілей захисту життя або здоров'я громадян, майна фізичних або юридичних осіб, державного або муніципального майна; охорони довкілля, життя або здоров'я тварин і рослин; запобігання діям, що вводять в оману споживача.

В Україні на виконання Державної програми стандартизації на 2006-2010 роки розроблено Програму перегляду чинних в Україні міждержавних стандартів (ГОСТ), розроблених до 1992 року, та приведення їх у відповідність до Угоди про технічні бар'єри в торгівлі Світової організації торгівлі.

Основними напрямками виконання даної програми є:

- перевірка міждержавних стандартів на відповідність законодавству, інтересам держави, потребам споживачів, рівню розвитку науки і техніки, вимогам міжнародних та регіональних стандартів, положенням Угоди про технічні бар'єри у торгівлі;
- перегляд міждержавних стандартів із внесенням змін до них, заміну їх на відповідні міжнародні або національні стандарти;
- скасування міждержавних стандартів, які втратили актуальність, не використовуються і не відповідають вимогам чинного законодавства.

Переліки міждержавних стандартів (ГОСТ), розроблених до 1992 року, чинність яких в Україні може припинитись, розміщуються на веб-порталі ДП «УкрНДНЦ» (http://www.ukrndnc.org.ua/) та публікуються в щомісячному інформаційному покажчику «Стандарти».

Міністерство економічного розвитку в грудні 2015 року скасувало 12 776 ГОСТів, які залишаться частково дійсними на перехідний період до 2018 року. Підприємства за бажанням зможуть застосовувати їх добровільно. Реформа системи технічного регулювання передбачає перехід від системи обов'язкових держстандартів до європейської моделі технічного регулювання, заснованої на застосуванні технічних регламентів і добровільному використанні стандартів.

Другим типом нормативних актів, з яких може бути отримана чинна

інформація для інженерних розрахунків, є Державні стандарти України (ДСТУ) – стандарти, розроблені відповідно до чинного законодавства України, що встановлюють правила загального і багаторазового застосування виробів, їх загальні принципи або характеристики, які стосуються діяльності чи її результатів, з метою досягнення оптимального ступеня впорядкованості. Вони розроблені на основі консенсусу та затверджені уповноваженим органом. Стандарти ДСТУ існують з 1993 року.

Станом на жовтень 2014 фонд національних стандартів в Україні становить 27,5 тис. документів, 7489 з яких гармонізовані з міжнародними та європейськими. Протягом 2014 прийнято 175 національних нормативних документів, з яких 115 гармонізовані з міжнародними та європейськими, підготовлено до затвердження 350 проектів стандартів.

Державні стандарти України розробляються на організаційно-методичні та загальнотехнічні об'єкти, а саме:

- організація проведення робіт із стандартизації, науково-технічна термінологія, класифікація і кодування техніко-економічної та соціальної інформації, технічна документація, інформаційні технології, організація робіт з метрології, достовірні довідкові дані про властивості матеріалів і речовин;
- вироби загальномашинобудівного застосування (підшипники, інструмент, деталі кріплення тощо);
- складові елементи народногосподарських об'єктів державного значення (банківсько-фінансова система, транспорт, зв'язок, енергосистема, охорона навколишнього природного середовища, оборона тощо);
- продукцію міжгалузевого призначення;
- продукцію для населення та народного господарства;
- методи випробувань.

Опишемо спосіб використання довідкової документації при вирішенні реальної виробничої задачі на прикладі обчислення ваги сортового металопрокату

«арматура» діаметром 80 мм довжиною 15 погонних метрів. Із проектної документації дізнаємось номер ГОСТу або ДСТУ. У нашому випадку це ГОСТ 5781-82, після цього знайти даний документ у паперовому вигляді шляхом перегляду змісту або за допомогою пошукових систем в електронному вигляді не складає ніяких труднощів. У таблиці ГОСТу знаходимо теоретичну масу 1 м профілю для даного діаметра – 39,46 кг. Виконанням арифметичної операції 15 * 39,46 дізнаємось, що вага 15 м арматури діаметром 80 мм становить 591,9 кг.

Наступний приклад виробничої задачі: потрібно дізнатись довжину 25 кг троса (ГОСТ 483-55) окружністю 65 мм. Для цього у вказаному ГОСТі знаходимо вагу 1 метра троса заданої окружності і формулу обчислення довжини: 25/0,32=78,125 м. У іншому випадку, при відомій довжині та необхідності дізнатись вагу, розрахунок був би іншим: 25*0,32=8 кг.

Приклад розв'язання іншої задачі за допомогою документації: на підприємстві є сталевий канат (6х36, 216 дротів, ГОСТ 7668-55) кількох різних діаметрів достатньої довжини; потрібно дізнатись чи зможе він витримати навантаження 5000 кг і якого діаметра повинен для цього бути, якщо буде експлуатуватись як у ручному режимі, так і в режимі підйому людей. Для вирішення такої задачі потрібно дізнатись коефіцієнт запасу міцності каната. Із документації дізнаємось, що для ручного режиму він становить 4,5, а для режиму підйому людей – 9. Оскільки канат буде працювати у двох режимах, то нам слід вибрати найбільший коефіцієнт. Помножуючи 5000 на 9 отримуємо вагу, при дії якої канат розірветься – 45000 кг. Із вказаного ГОСТу дізнаємось, що діаметр каната для підйому такої ваги має складати 28,5 мм.

Отримані результати подібних розрахунків використовуються при подальшому плануванні виробничого процесу.

1.3 Онлайн-системи інженерних розрахунків

Окрім програм, які є частиною операційної системи або створені іншими розробниками, для інженерних розрахунків можуть використовуватись різноманітні онлайн-сервіси. Щоб ними скористатись, потрібен лише доступ в

Інтернет та програма-браузер. Працюючи в інтерактивному режимі, такі обчислювальні системи не потребують встановлення на комп'ютер або смартфон. До недоліків можна віднести постійну необхідність інтернет-з'єднання, інколи більш високе навантаження на компоненти комп'ютера у порівнянні зі звичайними програмами. Легко отримати до них доступ можна, увівши необхідний запит у пошуковій системі (наприклад, «калькулятор ваги металу»).

Оглянувши велику кількість інтернет-ресурсів, які надають можливості інтерактивного виконання різних видів розрахунків, ми дійшли висновку, що їх можна умовно поділити на два види: загальноматематичні та специфічно інженерні.

До першої групи відноситься сайт ua.onlinemschool.com, створений для всіх, хто має справу з математикою. Він містить велику кількість калькуляторів для вирішення різноманітних математичних задач, які можуть використовуватись як школярами і вчителями, так і деякими інженерами на металообробних підприємствах. Сайт надає такі основні можливості:

- конвертування одиниць величин (маси і ваги, відстані та довжини, площі, об'єму, часу, швидкості, температури тощо);
- виконання будь-яких арифметичних операцій із позначенням виконаних дій;
- розрахунок відсотків;
- розв'язання рівнянь (квадратних, біквадратних, систем рівнянь методами Гауса, Крамера);
- обчислення n-того члена арифметичної прогресії та суми арифметичної прогресії;
- обчислення границі функції та похідної функції;
- розв'язання інтегралів;
- розв'язання задач з теорії ймовірності, комбінаторики та аналітичної геометрії
- робота з векторами та матрицями;

– обчислення площі, об’єму та периметра різноманітних геометричних фігур.

Проаналізувавши перелік даних можливостей, ми дізнались, що корисними для інженерів на цьому сайті є лише конвертери різноманітних величин, інтерактивні засоби розв’язання інтегралів, деяких геометричних задач та задач з теорії ймовірності. У багатьох випадках можливостей подібних сайтів недостатньо для автоматизації вирішення більшості інженерних задач.

Проте Інтернет містить надзвичайно велику кількість ресурсів для розрахунку ваги, довжини, площі та інших величин певних матеріалів, а також для виконання специфічних інженерних обчислень (таких як розрахунок кількості зварювальних матеріалів та газів для виконання ремонтних і виробничих робіт, необхідних розмірів для виготовлення втулки Гудрича (гумово-металевих підшипників) і т. п.). Одним із таких ресурсів є tda-spb.ru/calk-metall.html, який дозволяє дізнатись вагу будь-якого металу за вказаними розмірами. Проте найбільшу функціональність, на наш погляд, має сайт 1metal.com/calc-info.html, який надає користувачу більш гнучкий вибір параметрів для розрахунку. На відміну від попереднього ресурсу, він окрім виду металу дозволяє вибрати такі важливі для інженерів і планувальників параметри як вид прокату, марка сталі, густина тощо (довідкові дані для цього, напевно, були взяті з нормативних актів ГОСТ і ДСТУ).

Інколи перед інженерами виникає необхідність отримання результатів обчислень у різних системах числення з ціллю їх шифрування. Існує велика кількість онлайн-конверторів для подібних перетворень. Проте найбільше нам сподобався сайт numsys.ru, який дозволяє виконувати арифметичні операції додавання, віднімання, множення та ділення над цілими та дробовими числами в різних СЧ, а також перетворювати їх з однієї СЧ в іншу. До переваг можна віднести високу швидкість отримання точних результатів, до недоліків – можливість роботи лише з додатними числами та найбільша довжина чисел – лише 50 символів. Саме цей сайт використовувався нами при розробці програмного комплексу з подібними можливостями як еталон результатів.

Ми дійшли висновку, що при наявності відповідних технічних можливостей, здійснення розрахунків за допомогою онлайн-ресурсів сприяє більш швидкому та зручному вирішенню інженерних задач, хоча й існує проблема достовірності довідкових даних, які використовуються в подібних обчислювальних системах.

1.4 Обґрунтування потреби в новому програмному продукті

У даному розділі ми проаналізували існуючі способи виконання інженерних розрахунків на сучасних українських державних металообробних підприємствах. Нами було розглянуті усі класи ПЗ, які використовуються на підприємствах: як засоби для проектування та ведення бухгалтерського обліку, так і обчислювальні програми і системи пошуку довідкових даних, необхідних для здійснення розрахунків, описані апаратні аналоги програмних рішень та різноманітні онлайн-сервіси з подібним функціоналом.

Було виявлено, що у сферах інженерного проектування та планування ресурсів використовується достатньо якісне ПЗ від наших та закордонних виробників, яке надається на комерційній основі разом із технічною підтримкою, не існує ніяких проблем і з оформленням різноманітної документації. Проте часто перед інженерами та планувальниками ресурсів виникають задачі, рішення яких не можуть бути достатньо автоматизованими за допомогою існуючих стаціонарних програмних комплексів. Високий рівень інтерактивності та зручності розрахунків і отримання довідкових даних надають лише деякі онлайн-сервіси. Проте до ряду недоліків таких систем відносяться постійна потреба в інтернет-з’єднанні, яке присутнє не у всіх підрозділах виробничих комплексів, проблема достовірності довідкових даних і формул розрахунків, які використовуються в інтерактивному ПЗ. Як ми знаємо, помилки при розрахунках ваги, розмірів, навантажувальної здатності матеріалів і т. п. можуть привезти до великих фінансових збитків та пошкодження здоров’я робітників. Тому більшість інженерів та планувальників не довіряють покладені на них розрахунки автоматизованим онлайн-системам і віддають перевагу ручному пошуку

необхідної інформації для обчислень, що сприяє неекономічному використанню робочого часу. Крім того, такі сервіси не надають можливості працювати з числами довжиною 100-250 символів, що є необхідним в дуже рідких випадках.

Помічено, що великі машинобудівельні підприємства, які не є достатньо завантаженими роботою, взагалі не мають доступу до більшості сучасних технологій автоматизації розрахунків. На них до сих пір використовуються апаратні калькулятори та інколи навіть механічні обчислювальні пристрої.

Тому перед нами була поставлена задача розробити програмний комплекс, який усував би велику кількість недоліків існуючих аналогів, більшість з яких доступна тільки в онлайн-режимі. Нове програмне забезпечення для інженерних розрахунків повинно здійснювати обчислення високої точності (вміти працювати з числами великої довжини), мати високий рівень інтерактивності, надавати швидкий доступ до необхідних довідкових даних і можливості їх зручного поповнення та редагування, показувати всі виконані дії для їх контролю працівниками підприємства та бути доступним на комп’ютері з операційною системою Windows будь-якої версії без потреби в інтернет-з’єднанні.

РОЗДІЛ 2. ПРОГРАМНІ ТА АПАРАТНІ ЗАСОБИ ДЛЯ РОЗРОБКИ І ФУНКЦІОНУВАННЯ ПЗ

2.1. Програмно-апаратні вимоги

Звичайно, як і для всіх програм, для функціонування розробленого нами програмного комплексу підтримки інженерних розрахунків необхідний апаратний пристрій персональний комп'ютер (ПК) – персональна електронно-обчислювальна машина, яка має експлуатаційні характеристики побутового пристрою та універсальні функціональні можливості. Він використовується як засіб масової автоматизації в основному для створення автоматизованих робочих місць у соціальній та виробничій сферах діяльності, народному господарстві тощо. Комп'ютер експлуатується користувачами, які можуть не володіти спеціальними знаннями в галузі обчислювальної техніки та програмування.

Спочатку ПК був створений як обчислювальна машина, але пізніше почав використовуватись у інших цілях – як засіб доступу до інформаційної мережі, як мультимедійна та ігрова платформа. Сучасний ПК може використовуватися декількома людьми одночасно. Хоча з розвитком можливості такого пристрою розширюються, ПК все рівно буде вважатись персональним, не залежно від того, якими б обчислювальними можливостями він не володів. Якщо не так давно лише IBM-сумісні пристрої (однотипні комп'ютери) вважались ПК, то зараз будь-який комп'ютер, який використовується людиною за власними примхами, може розглядатись як персональний. Конструктивність на сьогодні не найважніший критерій ПК: практично будь-який комп'ютер може стати персональним у залежності від необхідності та можливостей його використання. Все більше користувачів використовують у якості ПК в основному його стаціонарні або портативні версії, різниця між якими полягає у швидкодії, габаритах та мобільності.

Отже, для розробленого нами програмного продукту потрібен звичайний ПК, який задовольняв би хоча б мінімальним вимогам операційної системи (ОС) – програмного комплексу взаємозв'язаних програм, призначених для управління ресурсами комп'ютера і організації взаємодії з користувачем. У логічній

структурі типової обчислювальної системи ОС займає положення між пристроями з їх мікроархітектурою, машинною мовою та, можливо, власними (вбудованими) мікропрограмами (драйверами) – з однієї сторони – і прикладними програмами з іншої. Операційна система дозволяє розробникам ПЗ абстрагуватись від деталей реалізації та функціонування пристроїв, надаючи мінімальний необхідний набір функцій. У більшості обчислювальних систем ОС являється основною, найбільш важливою (а іноді єдиною) частиною сімейств ПЗ. З 1990-х років найбільш вживаними ОС являються системи сімейств Windows, UNIX и UNIX-подібні системи.

Попередником операційних систем слід вважати службові програми (завантажники і монітори), а також бібліотеки часто використовуваних підпрограм, які почали розроблюватись з появою універсальних комп'ютерів першого покоління (кінець 1940-х років). Службові програми мінімізували фізичні маніпуляції оператора з пристроями, а бібліотеки дозволили запобігти багатократного програмування одних і тих самих дій (здійснення операцій вводу-виводу, обчислення математичних функцій і т. п.).

У якості операційної системи нами обрано сімейство Microsoft Windows. Даний вибір обумовлений тим, що сьогодні в світі близько 80% персональних комп'ютерів працює під керуванням тих чи інших версій Windows. Це призводить до поширення різноманітних зручних програм та середовищ розробки саме для платформи Microsoft. Не є виключенням і підприємство-замовник нашого програмного продукту: усі комп'ютери в ньому знаходяться під керуванням різних версій від Windows 98 до Windows 7. Така популярність пояснюється зручністю інтерфейсу користувача, підтримкою різноманітних сучасних технологій, більшості апаратних засобів комп'ютера – як внутрішніх, так і периферійних, а також багатьма іншими властивостями.

Для функціонування розробленого нами ПЗ, основною функцією якого є підтримка інженерних розрахунків на машинобудівному підприємстві, потрібен ПК, який задовольняв би наступним вимогам ОС Windows: процесор Intel Pentium з частотою 233 МГц (рекомендується не менше 300 МГц) або інший аналогічний

від фірми AMD, не менше 64 МБ оперативної пам’яті (рекомендується 128 МБ), жорсткий диск з вільним місцем не менше 20 МБ для роботи програми або 60 МБ для редагування існуючої програми за допомогою середовища розробки. Розроблена програма не потребує додаткового ПЗ, такого як .NET Framework або Microsoft Visual C++ Redistributable. Отже, для роботи та підтримки нашого продукту достатньо навіть дуже старого ПК з будь-якою версією ОС Windows.

2.2 Середовище розробки, платформа та мова програмування

Для створення програмного продукту було обрано інтегроване середовище розробки Lazarus останньої версії та мову програмування Object Pascal. Даний вибір був обумовлений вимогами підприємства у зв’язку з наявністю на ньому спеціалістів, які в майбутньому зможуть підтримувати та коригувати створену програму, наявністю готового програмного продукту, який був швидко та легко вдосконалений до необхідного рівня зручного комплексу для виконання інженерних розрахунків, а також особистим добрим володінням даної мови програмування.

Lazarus – це вільне (безкоштовне) середовище швидкої розробки ПЗ для компілятора Free Pascal, аналогічне Delphi. Даний проект базується на оригінальній багатоплатформній бібліотеці візуальних компонентів Lazarus Component Library (LCL), яка працює на багатьох операційних системах та апаратних платформах.

Free Pascal – це компілятор мов Pascal і Object Pascal, що функціонує під Windows, Linux, Mac OS X, FreeBSD, та іншими операційними системами. Таким чином, розроблені додатки можуть виконуватись практично на будь-якому комп’ютері під управлінням будь-якої системи.

Все що користувач бачить на екрані під час роботи різних програм, всі елементи (кнопки, бігунки, меню і т. п.) можна реалізувати в Lazarus, оскільки в даному середовищі використовується технологія візуального програмування. Для створення графічного інтерфейсу програми розробник використовує готові компоненти, значки яких знаходяться на відповідній панелі. Для новостворених

елементів програмний код генерується автоматично. Вручну програмуються тільки специфічні дії, які виконуватиме доданий елемент.

Перелічимо основні можливості середовища розробки Lazarus:

- підтримує перетворення проектів Delphi;
- реалізований основний набір елементів управління;
- редактор форм та інспектор об'єктів максимально наближені до середовища Delphi;
- вбудований налагоджувач;
- простий перехід для Delphi-програмістів завдяки подібності LCL до VCL;
- повністю юнікодний (UTF-8) інтерфейс і редактор, а тому відсутні проблеми з портуванням (переносом на іншу платформу) коду, що містить національні символи;
- потужний зручний редактор коду, що включає систему підказок, гіпертекстову навігацію по вихідних текстах, автоматичне завершення коду і рефакторинг;
- форматування коду «з коробки», використовуючи механізми Jedi Code Format;
- підтримка двох стилів асемблера: Intel і AT&T (підтримуються з боку компілятора);
- підтримка безлічі типів синтаксису Pascal: Object Pascal, Turbo Pascal, Mac Pascal, Delphi;
- має власний формат управління пакунками;
- автоматична збірка самого себе (під нову бібліотеку віджетів) натисненням однієї кнопки;
- підтримувані для компіляції ОС: Linux, Microsoft Windows (Win32, Win64), Mac OS X, FreeBSD, WinCE, OS/2.

До недоліків даного середовища розробки можна віднести:

- відсутність повної сумісності з Delphi (на відміну від Delphi, Lazarus

надає можливість створювати багатоплатформні програми);

- великий розмір скомпільованого файлу при стандартних налаштуваннях середовища (виправляється шляхом відключення можливостей зберігати у новоствореному файлі налагоджувальну інформацію);
- відсутність повноцінної документації (хоча документація на сам компілятор Free Pascal доступна онлайн, або в PDF- та HTM-документах, а документація на Lazarus доступна у вигляді Wiki-підручників, які можуть редагувати самі користувачі);
- відсутність повноцінної підтримки COM (реалізована тільки підтримка методів), оскільки сфера інтересів розробників Lazarus лежить в області багатоплатформного програмування, а не в області взаємодії лише з Windows-програмами;
- неможливість перегляду значення властивостей об'єктів під час налагодження, відображення тільки змінних і полів об'єктів.

При розробці віконної версії програмного комплексу підтримки інженерних розрахунків ми використовували такі основні можливості середовища Lazarus:

- редактор форм з можливістю додавання різноманітних елементів інтерфейсу користувача. Використовувались такі компоненти і можливості редагування їхніх властивостей: TLabel (текстова мітка довжиною в один рядок), TComboBox (список рядків з можливістю вибору рядку із випадного списку з постійним відображенням вибраного), поля вводу тексту TEdit та TMemo (TEdit дозволяє вводити та відображати лише один рядок, TMemo – декілька), прапорець TCheckBox (перемикач для візуалізації станів увімкнуто / вимкнуто), TButton (кнопки для запуску виконання певних дій);
- зручний текстовий редактор з можливістю підсвічування елементів програмного коду, ведення його «історії»;
- вбудований компілятор мови Free Pascal разом із здатністю візуального позначення помилок при збірці та виконанні програми.

Отже, при розробці програмного комплексу ми використовували лише незначну частину компонентів середовища Lazarus. Можливості автоматичного доповнення та рефакторингу коду, відображення спливаючих підказок назв процедур та функцій при написанні команд, збереження налагоджувальної інформації у скомпільованому файлі були вимкнуті.

2.3 Програми для додавання і редагування довідкових даних

Усі довідкові дані, які використовуються нашою програмою для розрахунків, зберігаються в текстовому форматі у вигляді txt-файлів, які можуть бути відредаговані та доповнені користувачами програми за допомогою будь-якого текстового редактора.

Текстовий редактор – це самостійна комп'ютерна програма або компонент програмного комплексу (наприклад, редактор вихідного коду інтегрованого середовища розробки або вікно введення в браузері), призначена для створення та зміни текстових даних взагалі та текстових файлів в окремому вигляді.

Текстові редактори призначені для роботи з текстовими файлами в інтерактивному режимі. Вони дозволяють продивлятись зміст текстових файлів та виконувати над ними різноманітні дії – вставку, видалення і копіювання тексту, контекстний пошук і заміну, сортування рядків, перегляд кодів символів і конвертацію кодувань, друк і т. п. Часто інтерактивні текстові редактори містять у собі додаткову функціональність, призвану автоматизувати дії по редагуванню (від записаних послідовностей натиснутих клавіш до повноцінних вбудованих мов програмування), або відображають текстові дані спеціальним чином (наприклад, з підсвіченням синтаксису).

Велика кількість текстових редакторів є редакторами вихідного коду, тобто вони орієнтовані на роботу з текстами програм на тих чи інших комп'ютерних мовах програмування. Види текстових редакторів:

– відрядковий (рядковий) (англ. line editor) - працює з текстом як з послідовністю пронумерованих рядків, виконуючи операції над текстом у вказаних рядках. Прикладом такого редактора є edlin, який входить до

складу MS-DOS;

- контекстний (рядковий) редактор (англ. context editor), прикладом якого може стати ECCE (англ. Edinburgh Compatible Context Editor), який виконує операції над текстом у поточній позиції.
- Екранний текстовий редактор - дозволяє користувачу переміщувати курсор у тексті за допомогою клавіш або інших пристроїв уведення. До таких програм можна віднести будь-який редактор, який працює в будь-якій сучасній віконній ОС.

До текстових редакторів з розширеною функціональністю відноситься текстовий процесор – це комп'ютерна програма, яка використовується для написання та модифікації документів, компоновки макету тексту і попереднього перегляду документів у тому вигляді, в якому вони будуть надрукованими. Сучасні текстові процесори, окрім форматування шрифтів і абзаців, перевірки орфографії включають можливості, які мали лише настільні видавничі системи, у тому числі створення таблиць і вставку графічних зображень. Найбільш відомими текстовими процесорами є Microsoft Word та OpenOffice Writer.

Операційна система Windows має вбудований текстовий редактор «Блокнот», який має лише базові можливості обробки тексту. Тому на підприємствах часто використовують розширений текстовий редактор Notepad++, який має більшу функціональність. Надаючи такі можливості як підсвічування синтаксису багатьох мов програмування і розмітки, нумерацію рядків, одночасна обробка декількох файлів (заміна і пошук тексту, збереження внесених змін), перегляд у різних кодуваннях, він сприяє більш зручній та швидкій обробці текстових даних.

Отже, існує безліч програм, які можуть використовуватись для зміни та доповнення довідкових даних: від найпростішого редактора, який входить до складу ОС, до потужних текстових процесорів. Проте здійснюючи такі операції користувачу слід пам'ятати, що вся інформація коректно оброблюється програмою тільки якщо вона впорядкована єдиним зрозумілим для неї чином. Наприклад, у деяких файлах прийнято, що найменування розташовуються до

символу «/» або «пробіл». При видаленні таких символів або доданні зайвих змінений текстовий файл може некоректно сприйматись нашою програмою та спричинити помилки. Або деякі компоненти програми використовують інформацію з декількох файлів, тому при її зміні або доповненні користувач повинен уважно слідкувати за відповідністю рядків, що зручно виконувати за допомогою спеціалізованих редакторів з функцією нумерації рядків. Видалення текстових файлів, які знаходяться в одному каталозі з програмою може призвести до пошкодження вже існуючих результатів розрахунків та помилок в роботі створеного нами ПЗ.

РОЗДІЛ 3. ПРОГРАМНИЙ КОМПЛЕКС ВИКОНАННЯ ІНЖЕНЕРНИХ РОЗРАХУНКІВ

Ціллю існування даного програмного комплексу є:

- виконання арифметичних операцій додавання, віднімання, множення, ділення цілих та дробових знакових чисел надвеликої довжини (більше 50 символів) у всіх можливих системах числення (2-36);
- конвертування цілих та дробових чисел із однієї системи числення в будь-яку іншу;
- зміна точності операції ділення (кількість знаків після коми);
- виконання інженерних розрахунків за заданими формулами та з використанням довідкових даних, які знаходяться в окремих файлах;
- ведення журналу здійснених операцій.

3.1 Принципи роботи програми та інтерфейс користувача

Програмний комплекс був розроблений за допомогою засобів вільно розповсюджуваного середовища Lazarus та мови Free Pascal для використання у віконному режимі, ґрунтується на принципах візуального, об’єктно-орієнтованого та процедурного програмування. Усі екранні елементи інтерфейсу користувача, як і довідкові дані, названі російською мовою. Їх можна умовно поділити на три групи:

- клавіши екранного вводу символів чисел та арифметичних операцій (символи вводяться в будь-яке активне змінюване поле;
- кнопки, поля вводу, випадні списки модулів інженерних розрахунків;
- допоміжні елементи (перемикачі, кнопки очищення різних полів, текстові мітки, поля операцій, «журналу» та результату обчислення).

Усі компоненти програми розташовані на одній формі, зовнішній вигляд якої представлено на рисунку 1.

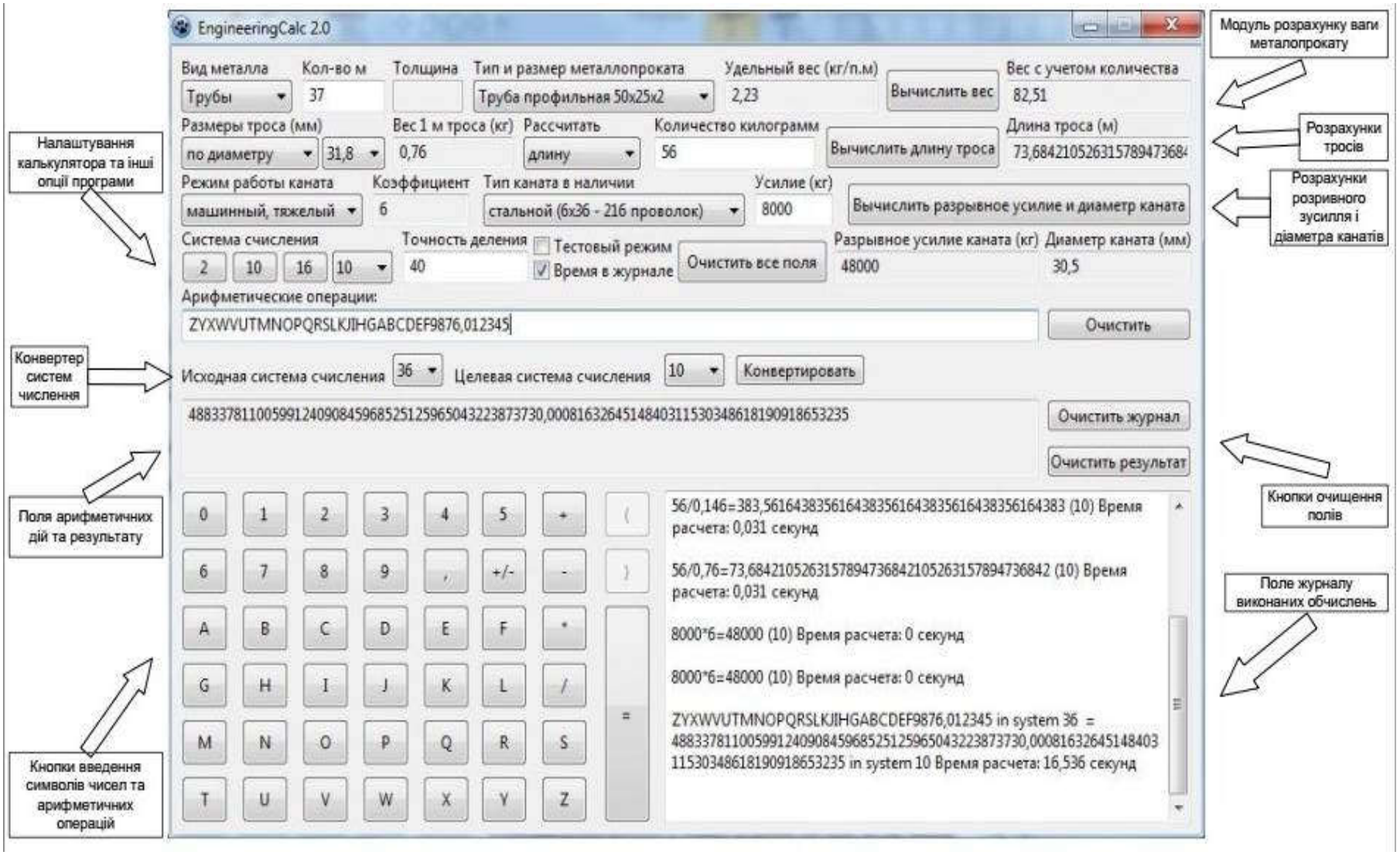

Рисунок 1 – Елементи інтерфейсу користувача

Отже, структуру даного програмного комплексу можна умовно поділити на три частини: калькулятор, конвертер систем числення та модулі інженерних розрахунків, які на даному етапі розвитку проекту містять можливості обчислення ваги металопрокату і троса, довжину троса, розривне зусилля та необхідний діаметр каната. Схему загального принципу роботи всіх компонентів програми проілюстровано на рисунку 2.

Рисунок 2 – Загальні принципи роботи ПЗ

Розглянемо особливості інтерфейсу користувача програми більш детально.

Кнопки введення числових символів активуються в залежності від того, в якій системі числення працює калькулятор. Отже, якщо калькулятор працює в шістнадцятковій системі числення, «екранна клавіатура» має вигляд, зображений на рисунку 3.

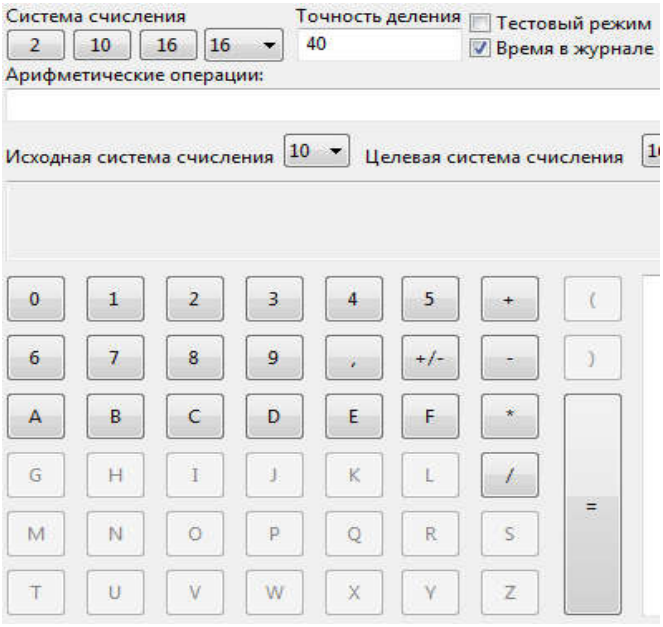

Рисунок 3 – Вигляд екранної клавіатури ПЗ в режимі роботи в 16-й СЧ

Кожна кнопка екранного вводу виконує дії, закодовані в єдиній, однаковій для всіх, процедурі:

procedure TForm1.NumberButton(Sender: TObject);

var ch:string;

begin ch:=(Sender as TButton).hint; charinput(ch); end;

У властивостях кожної такої кнопки в полі Hint (підказка) прописаний символ, який відповідає тому, який має вводитись при натисканні на кнопку. Дана процедура лише зчитує символ з поля підказки натиснутої кнопки і передає його для введення у відповідне поле за допомогою іншої процедури *charinput*. Така реалізація є найбільш раціональною та сприяє уникненню зайвого однотипного коду. У попередній версії програми для кожної кнопки екранного вводу числових символів була створена окрема процедура, яка відрізнялась від іншої лише символом, який має вводитись при натисканні.

Принцип роботи процедури *charinput* ґрунтується на використанні ГЗ типу boolean, які переходять у стан true при активації певного поля форми: підпрограма перевіряє стани всіх змінних і вводить символ у кінець поля, відповідного певній boolean-змінній. Оскільки після натискання на будь-яку кнопку інтерфейсу користувача фокус залишається на натиснутій кнопці, що не дуже зручно для користувача, дана процедура ще й повертає фокус на поле, в яке був уведений

символ та встановлює каретку в кінець рядка.

Подібним чином реалізована можливість активації та деактивації клавіш екранного вводу в залежності від вибраної СЧ калькулятора. Стан кнопок змінюється процедурою, в якій створюється масив елементів типу boolean та в залежності від номера активного чи неактивного елементу, активується або деактивується певна кнопка:

```
procedure tform1.buttonenable(n:byte); // n – система числення
var i:byte; a:array[0..35] of boolean;
begin
for i:=0 to 35 do a[i]:=false; // формування «списку» активних кнопок
for i:=0 to n-1 do a[i]:=true; // формування «списку» неактивних кнопок
if a[0]=true then b0.Enabled:=true else b0.Enabled:=false;
if a[1]=true then b1.Enabled:=true else b1.Enabled:=false;
if a[2]=true then b2.Enabled:=true else b2.Enabled:=false;
...
if a[35]=true then b35.Enabled:=true else b35.Enabled:=false; end;
```

Такий спосіб реалізації є набагато кращим аніж ручне формування переліку команд активації екранних клавіш для кожної СЧ.

Інтерфейс користувача також містить дві опції: «Тестовый режим» та «Время в журнале». Перша опція включає або виключає можливість виконання обчислень з використанням дужок. Оскільки ця можливість не в достатній мірі протестовані та потребує удосконалень, за замовчуванням вона вимкнена: на даному етапі розвитку проекту при виконанні пріоритетних дій у дужках можливі помилки (хоча відповідна процедура в повній мірі реалізована).

Друга опція корисна при тестуванні програми та необхідна для оцінки швидкодії обчислювальних алгоритмів: вона активує або деактивує можливість відображення кількості секунд та їх частин з точністю до трьох знаків після коми, витрачених на арифметичні операції або перетворення чисел з однієї СЧ в іншу.

До особливостей інтерфейсу користувача відноситься й автоматичне прокручування «журналу» виконаних операцій, наявність декількох кнопок

очищення різних полів, відображення спливаючих підказок при наведенні курсора мишки на перемикачі.

3.2 Модуль виконання арифметичних операцій

Арифметичні операції додавання, віднімання, множення та ділення є головними функціями даного програмного комплексу для виконання інженерних розрахунків. Як було сказано раніше, наша програма здійснює їх в усіх системах числення, з двійкової по тридцяти шісткову, над цілими та дробовими числами з урахуванням їх знаку, ґрунтуючись на принципах довгої арифметики.

Для збереження операндів у пам’яті та виконання різних операцій над ними використовується тип даних string (рядок), максимальна довжина якого у мові Pascal становить лише 255 символів. Проте використання новітнього середовища розробки Lazarus разом з директивою {$H+} збільшує найбільшу довжину рядка до 2 Гб. Отже, фізично наш програмний комплекс здатен виконувати обчислення над числами дуже великої довжини. Вибір такого типу даних для збереження операндів головним чином обумовлений наявністю різноманітних готових зручних функцій для обробки та перетворень введеної інформації, можливістю використовувати не тільки звичні для нас традиційні символи цифр, достатньою максимальною довжиною операндів.

Модуль виконання арифметичних операцій лише умовно може називатись модулем, та становить сукупність основних і допоміжних функцій для виконання розрахунків. Нічого спільного із терміном «модуль» він не має (традиційно модулем у програмуванні називають функціонально завершений фрагмент програми, оформлений у вигляді окремого файлу з вихідним кодом, призначений для використання в інших програмах), хоча частина створених нами процедур можуть бути легко вбудовані в інший програмний продукт.

Сутність роботи даного модуля ґрунтується на використанні як глобальних так і локальних змінних. Усі функції повертають лише одне значення, проте деякі з них зберігають дані і в ГЗ (хоча директива *var* при написанні аргументів функцій нами в деяких випадках використовувалась). У майбутньому планується

здійснення переробки коду програми, в ході якої будуть усунені всі ГЗ, використання яких вважаться ознакою поганого стилю програмування. Усі глобальні змінні прокоментовані у коді для більш кращого розуміння. У них зберігаються рядки вхідних операндів *s1* та *s2*, залишок від ділення *od*, «вирівняні» змінені рядки операндів *se1* та *se2*, система числення *sys*, точність операції ділення *prec* (кількість знаків після коми), позиції коми в рядках операндів *z1* та *z2*, прапорці *op_plusb* та *op_minusb*, за допомогою яких здійснюється фільтр дій, необхідних для виконання операцій лише множення та ділення.

3.2.1. Допоміжні функції калькулятора

Усі функції модуля виконання арифметичних операцій мають зрозумілу назву та прокоментовані у програмі для більш кращого розуміння коду. Їх можна умовно поділити на дві великі групи: основні та допоміжні. Допоміжні функції виконують дії, спільні для декількох арифметичних операцій, та використовуються головним чином для уникнення повторювальних дій. Це може частково утруднити розуміння коду програми іншими спеціалістами, але чіткі назви і наявність коментарів не дають втратити сутність змісту при розбиранні чужого коду. Перелічимо всі допоміжні функції, які використовуються для здійснення арифметичних операцій.

– *function ntoch(i:byte):string* – перетворює цифру на символ (наприклад, 5 – «5», 15 – «F» і т. д.). Вхідний параметр – число типу byte (найменший цілий тип), повертає результат у якості рядка (string). Принцип дії функції ґрунтується на перевірці умови для вибору однієї з двох операцій шляхом порівняння вхідного числа: якщо число більше 9, то до нього додається 55 (оскільки коди великих літер латинського алфавіту в таблиці символів починаються з 65) і функції присвоюється результат виконання команди *chr* (отримання символу за його кодом). В іншому випадку, якщо вхідне число менше 10, функція отримує значення вхідної цифри, перетвореної на рядок;

– *function chton(ch:char):byte* – функція обернена до *ntoch* (перетворює вхідний символ *ch* у ціле число типу byte). На початку підпрограми створюється множина символів *set1*, в яку за допомогою циклічної операції додаються символи цифр десяткової СЧ. Після цього здійснюється перевірка вхідного символу: якщо він входить у множину, то функція отримує значення його коду в таблиці символів шляхом виконання операції перетворення символу в число, в іншому випадку дана підпрограма повертає результат виконання команди *ord(ch)-55* (отримує числовий код символу). На відміну від попередньої функції *ntoch*, від отриманого коду символу необхідно відняти число 55, оскільки великі латинські літери в таблиці символів починаються з номера 65;
– *function normalize(var s1,s2:string):byte* – функція вирівнювання (або «нормалізації») вхідних операндів з комою, повертає позицію коми операнду з найбільшою цілою частиною. Результат роботи функції раніше використовувався у програмі для визначення позиції коми в результаті виконання деяких арифметичних операцій (зокрема додавання), але в ході більш детального тестування було, виявлено, що такий спосіб отримання необхідної позиції є правильним не в усіх випадках, тому він був замінений нами на інший більш надійний. На початку даної підпрограми у ГЗ *z1* та *z2* поміщуються значення позиції коми вхідних операндів (якщо операнд має лише цілу частину, то в змінну записується його довжина, збільшена на 1). Після цього здійснюється вирівнювання довжини цілої частини вхідних чисел: до числа з найкоротшою цілою частиною додається символ «0». Наступними діями є отримання довжин дробових частин операндів за допомогою виконання команди *length(s1)-pos(comma,s1)* (подібна команда виконується і для другого операнду); якщо кома відсутня, відповідній змінній присвоюється нульове значення. Результати виконання обох команд зберігаються відповідно в *z1* та в *z2*. Далі в кінець операнду з найкоротшою дробовою частиною додаються символи

«0» і коми видаляються з операндів. Перетворення операндів за допомогою функції *normalize* є необхідним при виконанні операцій додавання та віднімання, а також залежних від них операцій множення та ділення, над числами з різною довжиною цілої або дробової частин;

- *procedure eqsize(var s1,s2:string)* – процедура, подібна до вище описаної функції *normalize*. На відміну від попередньої, дана підпрограма здійснює «нормалізацію» лише цілих частин вхідних операндів та не повертає ніяких значень. Змінені операнди дублюються в глобальних змінних *se1* та *se*2.
- *function zerodelb(s:string):string* – функція видаляє зайві символи «0» на початку вхідного рядкового числа і повертає змінений рядок. Дана підпрограма працює лише з цілими числами. Принцип її дії ґрунтується на тому, що за допомогою циклічної команди *while s[i]='0' do inc(i)* (перед цим змінній *i* присвоєно значено 1) визначається кількість нульових символів на початку рядка, після чого символи видаляються командою *delete(s,1,i-1)*. Якщо після виконання даної операції символів у рядку не залишилось, функція повертає значення «0». Виконання даної функції є необхідним для обробки результату виконання арифметичних операцій, оскільки в ході них у більшості випадків залишаються зайві нулі;
- *function zerodele(s:string):string* – функція, подібна до попередньої, але видаляє зайві символи «0» в кінці рядка. Дана підпрограма виконує операції, протилежні функції *zerodelb*: спочатку змінній *i* присвоюється значення довжини вхідного рядка, після цього циклічною операцією *while s[i]='0' do dec(i)* здійснюється підрахунок символів «0» у вхідному операнді, які потім видаляються командою *delete(s,i+1,length(s)-i)*. Якщо результатами таких дій став пустий рядок, підпрограма повертає значення «0». Функція виконується лише для дробових чисел (перед її запуском здійснюється перевірка позиції коми в рядку);

– *function compare(s1,s2:string):char* – функція, яка порівнює між собою строкові числа *s1* та *s2* і повертає символ «>», «<» або «=». Якщо операнди рівні між собою функція отримує значення «=», в іншому випадку виконуються наступні дії: здійснюється нормалізація операндів за допомогою функції *normalize*, циклічна команда *while s1[i]=s2[i] do inc(i)* визначає позицію останньої рівної цифри, після чого шляхом порівняння кодів символів рядка в позиції *i* визначається знак, який повертає дана функція;

– *function check(s:string):boolean* – підпрограма перевірки відповідності рядкового числа заданій СЧ. Якщо операнд не правильний, функція повертає значення false, в іншому випадку отримує значення true. Спочатку здійснюється перевірка коректності вказаної в програмі СЧ (вона не повинна бути меншою 2 та більшою 36), вхідний операнд не може бути пустим. Після цього створюється множина символів, доступних лише у вказаній СЧ: циклом *for i:=48 to 48+sys-1 do set1:=set1+[chr(i)]* додаються символи, доступні в так званих «цифрових» системах числення. Якщо СЧ більша за 10, то циклом *for i:=65 to 65+sys-9 do* додаються символи латинських літер. Далі здійснюється перевірка вхідного рядка: якщо був знайдений символ, відсутній у множині, функції присвоюється значення false, цикл переривається і відбувається вихід з підпрограми, в іншому випадку функція завершує свою роботу із значенням true;

– *function sign(var s1,s2:string):boolean* – функція визначення знаку результату для операцій множення та ділення в залежності від знаків двох вхідних операндів *s1* та *s2* повертає значення true (результат від'ємний і до нього в майбутньому слід додати символ «-») або false (результат додатній і рядок результату ніяких змін не потребує). Підпрограма перевіряє перші символи обох операндів та визначає знак результату за традиційними правилами арифметики: якщо обидва операнди від'ємні, результат додатній, якщо один додатній і один

від'ємний, результат від'ємний, у іншому випадку результат додатній. Символи знаків видаляються із вхідних операндів, функція не додає знак до результату, а лише перевіряє початкові операнди перед виконанням арифметичним дій.

Кожна арифметична операція, за виключенням множення, реалізована декількома функціями, які отримують в якості вхідних операндів два рядки *s1* та *s2* і повертають результат також рядкового типу. Тепер опишемо механізми здійснення всіх арифметичних операцій у різних СЧ.

3.2.2 Арифметичні операції додавання та віднімання

Операція додавання реалізована за допомогою двох функцій: *plus* та *op_plus*. Перша з них є основною і включає в себе лише дії для перевірки коректності введених даних і визначення знаку результату. На її початку виконується перевірка коректності операндів за допомогою вже описаної нами функції *check*, при невідповідності хоча б одного з двох чисел підпрограма завершується зі значенням *Enter correct operands!*. Далі слідують операції визначення знаку результату, і в залежності від нього здійснюється вибір допоміжних функцій – *op_plus* (додавання чисел без знаку) або *op_minus* (віднімання чисел без знаку). Звичайно, перед їх викликом видаляється знак числа, якщо воно від'ємне. Вибір функцій *op_plus* або *op_minus* при виконанні операції *plus* здійснюється за такими принципами:

- якщо обидва числа від'ємні або обидва додатні, викликається підпрограма додавання без знаку *op_plus*;
- якщо одне з чисел від'ємне, а інше – додатне, операнди віднімаються функцією *op_minus*.

Функція *op_plus* виконує додавання як цілих так і дробових чисел. На її початку визначається позиція коми вихідного результату додавання шляхом визначення найдовшої дробової частини одного з двох операндів. Кількість знаків після коми першого операнду записується командою *length(s1)-pos(comma,s1)* у ГЗ *z1*, другого операнду – відповідно командою

length(s2)-pos(comma,s2) у змінну *z2*. В іншому випадку, при відсутності дробових частин, відповідні *z1* та *z2* отримують нульові значення. Після цього найбільше із значень, тобто *z1* або *z2*, присвоюється локальній змінній цілого типу *ср*. Далі відбувається нормалізація операндів за допомогою вже описаної функцій *normalize* та процедури *eqsize*, коми видаляються і подальші операції виконуються як над цілими числами. Механізм виконання арифметичної операції додавання наступний:

1. Ініціалізація: змінним *p* та *q* присвоюються довжини вхідних операндів відповідно *s1* та *s2*; у змінній *i* зберігається кількість «десятків» (вона залежить від СЧ), яку необхідно додати до результату *r* (початкові значення *i* та *r* – 0);
2. Виконання циклу «*while (p>0) and (q>0)*» («поки не будуть оброблені всі символи обох рядків»), у ході якого здійснюється наступна послідовність дій: за допомогою функції *chton* отримуються числові значення символів операндів *s1* та *s2* з індексами *p* та *q* і зберігаються у змінних *c* (для рядка *s1*) та *d* (для рядка *s2*), після чого *p* та *q* зменшуються на 1; далі вони разом з десятком *i* додаються до результату *r* за допомогою звичайної стандартної операції додавання (команда *r:=c+d+i*); виконується перевірка: якщо результат *r* більший за СЧ, яка менша на 1, то від результату віднімається вказана СЧ і кількість «десятків» *i* стає 1, в іншому випадку змінна *i* отримує нульове значення; за допомогою функції *ntoch* визначається символ результату *r* і додається до кінцевого рядка *e*.

Після виконання циклу відбувається перевірка, чи є «залишок» *i* (якщо є, то на початок кінцевого результату додається символ «1», який збільшує число на один розряд) і видалення зайвих символів «0» за допомогою команди *zerodelb*. Наступні дії пов’язані лише з роботою із комою та виконуються у випадку якщо прапорець *op_plusb* має значення false (якщо дана функція *op_plus* викликається через підпрограму множення, а не тільки для одноразового додавання чисел, *op_plusb* отримує значення *true*). Це пояснюється тим, що кома, поставлена

процедурою додавання, зовсім не потрібна у результаті для операції множення, оскільки механізми визначення позиції коми в цих операціях різні. Отже, після встановлення коми за допомогою команди *insert(comma,e,length(e)-cp+1)* в уже вище визначеній позиції *cp* відбувається доробка результату: якщо кома є першим символом, до результату додається «0», якщо останнім – кома видаляється; у випадку коли після всіх цих дій кома відсутня, здійснюється повторний виклик функції *zerodelb*, інакше – видалення зайвого символу «0» здійснюється обома командами *zerodelb* та *zerodele*. Після цього функція отримує значення рядка *e*.

Функція *minus* виконує операцію віднімання цілих та дробових знакових чисел. Принцип її дії подібний до операції додавання: на початку виконується перевірка коректності вхідних операндів, після чого визначається знак результату і викликаються відповідні підпрограми за певних умов:

- якщо обидва числа від'ємні або обидва додатні, вони віднімаються функцією *op_minus*;
- якщо одне число від'ємне, а одне додатне, операнди додаються за допомогою підпрограми *op_plus*, причому якщо перше число від'ємне, а друге додатне, то результат від'ємний.

На початку процедури *op_minus* здійснюється ініціалізація локальних змінних: *mc* типу boolean, яка відповідає за знак результату, отримує значення false, а цілочислова змінна *cp* (позиція коми) – значення 0. Подібно до вище описаної функції додавання, операція ділення використовує дану функцію віднімання, але деякі її команди є для неї зайвими. Тому вони фільтруються за допомогою ГЗ типу boolean *op_minusb*: функція порівняння чисел *compare* запускається тільки якщо *op_minus* викликається не через функцію ділення (*op_minusb* у стані false). Таке порівняння необхідно для визначення того факту, чи не віднімаємо ми від меншого числа більше: якщо перший операнд менший, а другий більший, вони міняються місцями за допомогою локального рядка *t* і змінна *mc* отримує значення true (після виконання арифметичних дій результат стане від'ємним і необхідно додати до нього символ «-»).

Подальші операції здійснюються для визначення позиції коми в результаті

віднімання. Спочатку в глобальну змінну *z1* записується результат роботи команди *length(s1)-pos(comma,s1)* (якщо кома відсутня, *z1* отримує нульове значення), подібною операцією визначається зміст *z2*. Якщо *z1* більша за *z2*, змінній *cp* присвоюється значення *z1*, в іншому випадку – значення *z2*. Після таких дій виконується «нормалізація» (вирівнювання) операндів за допомогою вже описаних підпрограм *normalize* та *eqsize*, у ході яких коми видаляються. Далі слідують команди ініціалізації необхідних для циклу віднімання змінних цілого типу integer: у *n* записується довжина рядка *s1* (означає загальну кількість ітерацій циклу), у *r* (результат віднімання окремих розрядів) та у *i* («позичена одиниця» правішого розряду) – 0.

Наступний алгоритм, реалізований нами у циклі «*while (n>0)*», повністю ідентичний способу віднімання чисел у стовпчик:

1. цілочислові змінні *k* та *p* отримують числові значення *n*-х символів відповідно операндів *s1* та *s2* за допомогою допоміжної функції *chton*, причому від *k* ще й віднімається «позичена одиниця» *i* та кількість ітерацій циклу *n* зменшується на 1;
2. подальші дії залежать від числа *k*: якщо воно дорівнює -1, то йому присвоюється значення *sys-1*, де sys – СЧ роботи калькулятора; якщо *k* навпаки більша за *p*, то до неї додається вказана СЧ та змінній *i* присвоюється значення 1, в іншому випадку *i* отримує значення 0, яке означає, що позичення розрядів не виконувались;
3. результату віднімання *r* присвоюється різниця змінних *k* та *p* (команда «r:=k-p;») і в кінець рядка кінцевого результату *s* додається число *r*, перетворене на символ допоміжною функцією *ntoch*.

Після завершення циклу виконується доробка результату *s*: запускається допоміжна функція *zerodelb* для видалення зайвих символів «0» на початку рядка, якщо вони є та результат має бути цілим числом. Подальші команди виконуються, коли ГЗ *op_minusb* знаходиться в стані false (підпрограма віднімання *op_minus* викликана саме користувачем, а не операцією ділення): стандартна процедура *insert* вставляє кому в позицію *length(s)-cp+1*, циклічно видаляються зайві

символи «0» і коми з обох боків кінцевого рядка *s*, при необхідності додається невистачаючий «0» і змінюється знак результату. У кінці функції присвоюється значення змінної *s*.

3.2.3 Арифметичні операції множення та ділення

Підпрограма *mult* здійснює множення цілих та дробових знакових чисел у всіх допустимих СЧ. Подібно до функцій вище описаних арифметичних операцій, на її початку здійснюється перевірка коректності введених даних та ініціалізація ГЗ і ЛЗ. Глобальна змінна *op_plusb* отримує значення true, яке означає, що дана функція викликатиме підпрограму додавання *op_plus*, яка повинна повернути суму чисел без урахування коми та знаку. Локальна змінна *ms*, «відповідальна» за знак числа, ініціалізується із значенням false, після чого, якщо хоча б одне з чисел від'ємне, їй присвоюється відповідне значення допоміжною вже описаною нами функцією *sign*. Далі визначається позиція коми результату множення: у глобальну цілочислову змінну *z1* заноситься результат роботи команди *length(s1)-pos(comma,s1)*, який визначає кількість знаків після коми першого операнду; аналогічна команда виконується для другого операнду: *z2:=length(s2)-pos(comma,s2)*. ЛЗ *n1*, в якій зберігається позиція коми результату множення, отримує значення суми *z1* та *z2*, після чого коми видаляються з операндів (при їх наявності). Якщо першим символом операнду є «0», то здійснюється видалення зайвих символів за допомогою функції *zerodelb*.

Перед самим циклом множення виконується вирівнювання операндів процедурою *eqsize* та починається ініціалізація необхідних змінних: у *n* заноситься довжина першого операнду *s1*, у *l1* – довжина другого *s2*, змінні *r* (результат звичайного множення розрядів), *i* та *i1* (кількість символів «0», які необхідно додати до суми результатів відповідно першого та другого циклів) отримують значення 0; результат всієї операції множення зберігається в рядку *p2*, який перед запуском циклу приймає значення «0».

Реалізований нами алгоритм множення нагадує спосіб виконання даної арифметичної операції у стовпчик. Основний цикл перетворює усі розряди

першого операнду, використовуючи результат роботи другого вкладеного циклу, який проходить по всім розрядам другого операнду. Головний цикл «*while l1<>0*» починається з ініціалізації змінних, необхідних для роботи другого вкладеного циклу: змінна *b* типу byte отримує числове значення *l1*-го символу числа *s2* за допомогою функції *chton*; змінній *n* присвоюється довжина рядка *s1*; сума розрядів зберігається в рядковій змінній *p*, яка перед запуском другого циклу отримує значення «0»; цілочисловий лічильник кількості символів «0» *i*, необхідний у другому вкладеному циклі також ініціалізується із значенням 0; значення кількості кроків основного циклу *l1* зменшується на 1, після чого починає роботу другий вкладений цикл «*while n>0*».

У ньому виконуються дії в такій послідовності:

1. цілочисловій змінній *a* присвоюється результат роботи допоміжної функції *chton* аргументом якої є *n*-й символ числа s1 (*a:=chton(s1[n])*);
2. зменшується кількість кроків *n* на 1;
3. у змінну *r* потрапляє добуток чисел *a* та *b*;
4. змінна *c* отримує результат цілочислового ділення *r* на систему числення *sys*;
5. від числа *r* віднімається добуток *c* на *sys* і присвоюється змінній *d*: *d:=r-(c*sys)*;
6. у рядок *e* потрапляє конкатенація двох рядків, отриманих у результаті обробки командою *ntoch* чисел *c* та *d*: *e:=ntoch(c)+ntoch(d)*;
7. збільшується кількість символів «0», які необхідно додати до рядка *e* (змінна *i*);
8. якщо *i* більша за 0, виконується цикл *for j:=1 to i-1 do e:=e+'0'*;
9. у рядковій змінній *p* нарощується сума чисел *p* та *e* шляхом виконанням команди *op_plus(p,e)*;
10. завершення вкладеного циклу і продовження виконання основного.

Після перелічених дій у головному циклі здійснюються наступні операції: збільшується кількість символів «0», які необхідно додати в кінець суми *p*, отриманої у вкладеному циклі (змінна *i1*); якщо число *i1* більше за 0, до суми *p*

циклічно додається символ «0» у кількості *i1*-1; рядкова змінна *p2* отримує суму чисел *p2* та *p*:, тобто суму результатів обох циклів.

По завершенню роботи достатньо складного циклу множення виконується доробка результату *p2*: якщо його довжина менша або дорівнює сумі кількості знаків після коми обох операндів *n1*, отриманій перед основним циклом множення, то на початок результату додаються символи «0» у кількості *n1-length(p2)*; якщо *n1* більша за 0, командою *insert(comma,p2,length(p2)-n1+1)* у результат вставляється кома; при наявності дробової частини видаляються зайві символи «0» у кінці результату; видаляється кома, якщо вона є останнім символом у *p2*, і визначається знак результату за допомогою змінної *ms* типу boolean; функції *mult* присвоюється значення рядка *p2*.

Операція ділення реалізована за допомогою трьох функцій: *divide* (ділення з остачею), *edivide* (ділення у стовпчик) та *dividewr* (ділення без остачі). Головною з них є *divide*, яка циклічно викликає *edivide*, яка в свою чергу використовує *dividewr*. Почнемо опис реалізації операції ділення надвеликих цілих та дробових чисел у всіх СЧ з найголовнішої функції *divide*.

Як прийнято при реалізації подібних операцій на початку основної підпрограми ділення виконується перевірка коректності вхідних операндів функцією *check* та ініціалізуються необхідні змінні: глобальна *op_minusb* отримує значення true (при операції віднімання не використовуватиметься можливість обробки дробових операндів); якщо хоча б одне з чисел від'ємне, у змінну *ds* типу boolean записується результат роботи допоміжної функції обробки знаку *sign* (в іншому випадку *ds* дорівнює false).

Перед виконанням основного циклу ділення дробові числа оброблюються нами наступним чином:

- у ГЗ *z1* та *z2* записується кількість знаків після коми обох операндів відповідно *s1* та *s2*: *z1:=length(s1)-pos(comma,s1)*;
- найбільша з двох кількостей зберігається в змінній *pc*;
- виконуються дії, щоб позбутися коми в обох або в одному з чисел: формується рядок *dp* із початковим змістом «1», в кінець якого циклом

for додаються символи «0» у кількості *pc*, отриманій на попередньому кроці; обидва числа перемножуються з рядком *dp*, у результаті чого вони стають цілими числами.

Після завершення функцією перелічених підготовчих операцій виконуються дії для формування результату ділення із заданою точністю: у рядок кінцевого результату *x* записується результат роботи функції ділення у стовпчик *edivide*, після чого в його кінець додається кома, та до глобальної змінної *od*, яка використовувалась попередньою функцією для збереження остачі від ділення, додається символ «0»; виконується головний цикл ділення «*while (prec>0) and (od<>'00')*» («поки не буде досягнута задана користувачем точність та остача не нульова»), який включає в себе наступні дії:

1. частка від ділення *od* на друге число *s2* заноситься в рядок *r*: *r:=edivide(od,s2)*;
2. рядок кінцевого результату *x* поєднується з рядком *r* (*x:=x+r*);
3. до остачі від ділення *od* додається символ «0»;
4. задана точність *prec* зменшується на 1.

Далі виконується доробка результату: при необхідності видаляється зайва кома, якщо вона є останнім символом, додається знак в залежності від стану вже описаної змінної *ds*, видаляються зайві символи «0» в дробовій частині рядка результату.

Зупинимось більш детально на функції ділення в стовпчик *edivide*, яка використовується головною підпрограмою *divide*. На її початку здійснюється перевірка, чи не дорівнює один з операндів «0», оскільки на нуль ділити не можна. Наступний цикл формує рядок *o* із символів першого операнду *s1*, доки число, сформоване в ньому не буде більшим або рівним другому операнду *s2* та не будуть оброблені усі символи *s1*. Якщо довжина першого операнду *s1* дорівнює довжині другого операнду *s2*, рядковій змінній результату *r* присвоюється частка від цілочислового ділення *dividewr s1* на *s2*, в іншому випадку виконується цикл, перед яким встановлюється початкова кількість оброблюваних символів *i* на 1 та рядку *o* присвоюється перший символ операнду

s1. Отже, цикл «*while i<length(s1)+1*», який імітує ділення у стовпчик, включає в себе наступні дії:

1. число *o* ділиться на число *s2* за допомогою функції *dividewr* і частка зберігається в рядку *t*: *t:=dividewr(o,s2)*;
2. до рядка результату *r* додається отриманий рядок *t*: *r:=r+t*;
3. у *t* записується добуток останнього числа майбутньої частки *r* на операнд *s2*: *t:=mult(r[length(r)],s2)*;
4. від числа *o* віднімається число *t*: *o:=minus(o,t)*;
5. перевірка рядка *o*: якщо на попередньому кроці був отриманий нульовий результат, рядок *o* стає пустим: *if o='0' then o:=''*;
6. перехід на наступний символ шляхом збільшення лічильника *i*;
7. до рядка *o* додається *i*-й символ операнду *s1*.

Після завершення даного циклу або в іншому випадку після отримання частки за допомогою функції *dividewr* при необхідності виконується видалення зайвих нулів на початку рядка результату *r* за допомогою функції *zerodelb*. Підпрограма edivide повертає вміст змінної *r*.

Функція ділення чисел без остачі *dividewr* має дуже коротку реалізацію та базується на простому циклі «*while (compare(s1,s2)='>') or (compare(s1,s2)='=')*» («поки перший операнд більше другого або дорівнює йому»): на кожному кроці від першого числа *s1* віднімається друге *s2* і рядковий лічильник *r* збільшується на 1 за допомогою команди *op_plus(r,'1')*; після виконання циклу глобальній змінній *od*, в якій зберігається залишок від ділення, присвоюється значення рядка *s1*; якщо в ході циклічних перетворень рядок цілочислової частки *r* виявився пустим, його вміст змінюється на «0».

Описаний спосіб ділення довгих чисел у всіх СЧ має один серйозний недолік: частка дуже великих чисел обчислюється занадто довго. На відміну від інших арифметичних операцій, час її розрахунку може досягати кількох хвилин. Наприклад, при тестуванні помічено, що ділення між собою двох 34-символьних чисел тривало близько двох хвилин, а усі інші операції виконувались набагато швидше (наприклад, операція множення тих самих чисел

виконувалась менше однієї секунди). Ми припускаємо, що така низька швидкодія спричинена не дуже вдалим способом отримання цілої частини за допомогою функції *dividewr*. Не дивлячись на свою дуже коротку реалізацію, вона циклічно повторює дуже велику кількість однакових трудомістких дій, особливо це помітно при діленні дуже довгого числа на дуже коротке. Тому операція ділення є найповільнішою з усіх інших і потребує повної переробки для досягнення необхідної швидкодії для роботи з надвеликими числами.

Для тестування усіх арифметичних операцій нами була створена консольна програма, в якій виконуються усі дії за допомогою вище описаних підпрограм, помічається загальна тривалість обробки однакових операндів чотирма алгоритмами. Результати виконання арифметичних операцій, реалізованих нами мовою програмування Pascal, повністю співпадають з результатами майже аналогічного онлайн-калькулятора numsys.ru, який здійснює ті самі операції, але працює з числами не довше 50 символів. Тому числа, довше 50 символів нами не випробувались, оскільки не було знайдено аналогічного ПЗ для порівняння. Результати виконання обчислень представлені на рисунку 4, при їх порівнянні слід обов’язково звертати увагу на вказану СЧ («*Number system*») та кількість знаків після коми операції ділення («*Divide precision*»).

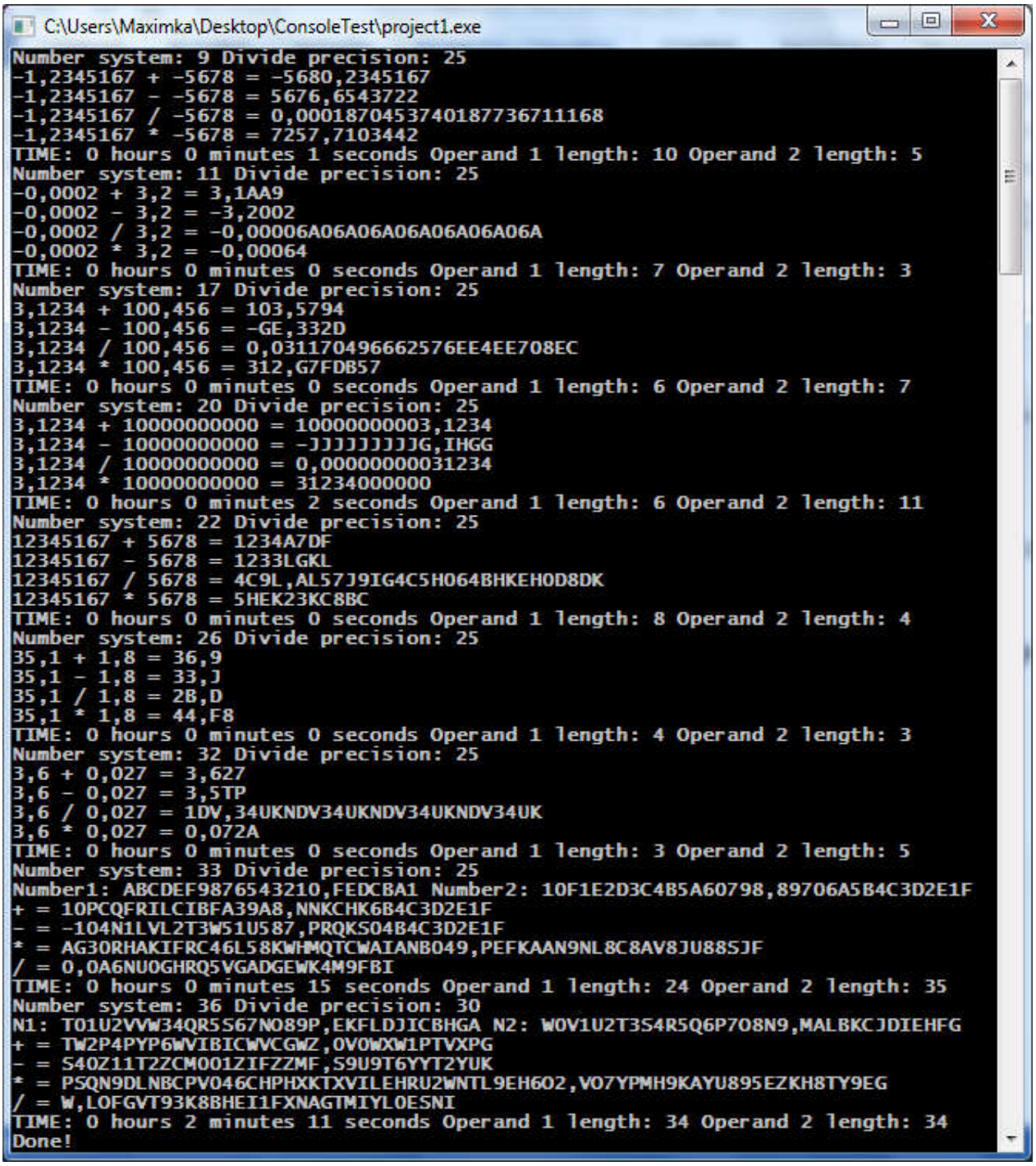

Рисунок 4 – Результати виконання арифметичних операцій у різних СЧ.

3.2.4. Обробка арифметичних виразів

Калькулятор містить також дві функції для обробки арифметичних виразів, які складаються з кількох дій. Перша з них *strproc* виділяє операції в дужках і передає їх у другу функцію *strprocwb*, яка послідовно виконує арифметичні дії без дужок.

Обробка дужок здійснюється циклом *while*, який працює поки рядок вхідного виразу *s* не пустий. Результат виконання обчислень без дужок зберігається у змінній *t2*, яка на кожному кроці стає пустою. Якщо був знайдений символ відкриваючої дужки, у цілочислову змінну *p* поміщується її позиція, у рядок *t1* потрапляють усі символи перед дужкою шляхом виконання команди *copy(s,1,p-1)*, і скопійований вираз видаляється із вхідного рядка разом із відкриваючою дужкою. Подібним чином додаються символи перед закриваючою дужкою в рядок *t2*. Після цього циклом *for* здійснюється пошук дужок у *t2*, вони видаляються і отриманий арифметичний вираз обчислюється іншою функцією *strprocwb*. До рядка кінцевого виразу *r* додається зміст *t1*, який містить числа та дії за межами дужок, та *t2*, який містить результат обчислень у дужках. Після завершення таких циклічних перетворень і отримання результатів усіх операцій у дужках здійснюється обчислення виразу *r* функцією *strprocwb*.

Дана функція знаходиться на стадії тестування, при її роботі можливі помилки (особливо при виконанні операцій «великої вкладеності»), тому за замовчуванням можливість вводити символи дужок відключена у програмі (їх можна задіяти, активувавши перемикач «Тестовый режим» у вікні користувача). Підпрограма не включає в себе перевірку коректності усього арифметичного виразу (якщо користувач введе неправильний вираз для розрахунку або помилково не поставить дужку, калькулятор може невірно його розрахувати або видати повідомлення про помилку).

Функція обробки арифметичних виразів без дужок *strprocwb* використовується для обчислень коректно введених арифметичних виразів, які складаються з більше ніж двох операндів. З її допомогою можна обчислювати вирази, які складаються з цілих та дробових знакових чисел і мають вигляд *a+b-c* або *a+-b*c/-d* (тобто припускається, що користувач увів від'ємний знак числа прямо після символу арифметичної операції без дужок). Дана функція має більш складний принцип дії, аніж *strprocwb*.

Перед основним циклом формування результату здійснюється ініціалізація змінної *b* типу boolean, призначеної для встановлення знаку числа; цілочислова

змінна *pr* приймає значення вказаної користувачем точності ділення (ГЗ *prec*) для її майбутнього відновлення, оскільки у процесі розрахунків *prec* зменшується; здійснюється пошук і видалення всіх дужок; формується множина *set1*, до якої додається символ коми, а також цифри вказаної користувачем СЧ способом, висвітленим при описі допоміжної функції *check*; здійснюється перевірка першого символу вхідного рядка: якщо він дорівнює «-», *b* отримує значення true та знак видаляється; формується перший операнд *s1*, до якого додаються лише ті символи, які входять у множину *set1* (це означає, що якщо користувач увів число, яке є некоректним для вказаної СЧ, воно буде сформовано не повністю, що може спричинити невірний результат); якщо *b* має позитивне значення, до першого операнду додається знак, і сформоване число видаляється з вхідного рядка. Основний цикл обчислення арифметичного виразу виконується доки рядок вхідного виразу не пустий, включає в себе дії у такій послідовності:

1. змінна *b*, яка «відповідає» за знак, отримує значення false;
2. на початку кожної ітерації циклу глобальна змінна *prec* (кількість знаків після коми у результаті роботи операції ділення) відновлюється із вже описаної *pr*;
3. глобальні *op_plusb* та *op_minusb* отримують значення false;
4. очищується рядок другого числа *s2*;
5. у змінну *c* типу char записується перший символ вхідного рядка всього виразу, який означає арифметичну операцію, необхідну для виконання, після чого він видаляється;
6. формується число *s2* разом із його знаком шляхом виконання вже вище описаних операцій для *s1*, яке видаляється із основного рядка;
7. після отримання необхідного другого операнду виконується арифметична операція, вказана символом *c* та її результат циклічно оновлюється у змінній *s1*.

Після завершення циклу функція *strprocwb* отримує значення рядка *s1*.

На даному етапі розвитку проекту описана підпрограма має один серйозний недолік: відсутня послідовність команд перевірки коректності введеного виразу.

У випадку введення чисел з непридатними символами операнди отримують лише цифри перед першим неправильним розрядом, а решта символів просто ігнорується, це може спричинити некоректність результату, про що користувач не буде повідомлений. На щастя, екранна клавіатура дозволяє вводити лише ті символи, які є доступними у вибраній СЧ, що дозволяє частково уникнути помилок при формуванні арифметичного виразу користувачем. У майбутньому планується вдосконалення даної функції.

3.3 Конвертування чисел з однієї системи числення в іншу

Перетворення цілих та дробових додатних чисел з однієї СЧ в іншу здійснюється за допомогою функцій вище описаних арифметичних операцій. Воно реалізоване нами за допомогою двох підпрограм: основної *numberconvert* та допоміжної *convertfromdec*. Перша з них переводить числа у десяткову СЧ за такою формулою: $A_{10}=a_{n-1}*q^{n-1}+...a_0*q^0+a_{-1}*q^{-1}+...+a_{-m}*q^{-m}$, де a – розряд числа, n – його порядковий номер, q – початкова СЧ; для розрядів цілої частини n додатне, для розрядів дробової частини n від'ємне. Друга функція *convertfromdec* перетворює числа з десяткової СЧ у будь-яку іншу (не меншу за 2 та не більшу за 36), для цілої частини послідовно повторюючи операцію її ділення на СЧ певну кількість разів, поміщуючи на початок рядка результату його залишки у вигляді відповідних символів; дробова частина перетворюються навпаки шляхом її множенням на СЧ і результат формується з цілих частин добутків. Окрім усіх процедур, функцій та глобальних змінних калькулятора конвертер систем числення додатково використовує глобальні цілочислові змінні *sys1* та *sys2* для запам'ятовування відповідно вхідних та цільових СЧ, а також додаткову допоміжну функцію *power* для піднесення розрядів чисел у степінь з цілим додатним або дробовим показником.

Допоміжна функція *power* має коротку просту реалізацію, у якості вхідних параметрів вона отримує два рядки: *s* (вхідне число) та *p* (показник степеня, який завжди знаходиться в десятковій СЧ). На її початку ініціалізується змінна *b* типу boolean, яка отримує значення true, якщо степінь від'ємний (у цьому випадку

символ «-» видаляється з рядка *p*). Початковий рядок результату *t* приймає значення «1», оскільки добуток не може починатись з порожнього рядка або з 0. Після цього виконується цикл *for i:=1 to strtoint(p) do t:=mult(s,t)*, в якому вхідне число *s* множиться само на себе *p* разів. Якщо змінна *b* отримала позитивне значення, число 1 ділиться на отриманий результат для піднесення у від'ємний степінь. У якості результату функція повертає рядок *t*.

Зупинимось більш детально на підпрограмі конвертування чисел з десяткової СЧ у будь-яку іншу *convertfromdec*, яка працює з точністю, вказаною користувачем у відповідному полі з назвою «Точность деления». Перед початком двох основних циклів перетворень виконуються наступні дії:

- ГЗ *sys* отримує значення 10, це означає, що усі арифметичні операції будуть виконуватись лише у десятковій СЧ;
- змінна *pt* запам'ятовує точність ділення *prec* для її майбутнього відновлення;
- рядковий залишок від ділення *od* ініціалізується із значенням «0»;
- допоміжні рядки *r*, *t*, *t1*, *fs* отримують пусті значення;
- при відсутності дробової частини у змінну *fs* потрапляє весь вхідний рядок, в іншому випадку ціла частина виділяється командою *copy(s,1,pos(comma,s)-1)* та у вхідному рядку вона замінюється на «0».

Цикл конвертування цілої частини виконується поки не буде досягнута задана точність та змінна *fs* не отримає значення «0». Він включає в себе такі команди: у рядку *fs* на кожному кроці зберігається частка від ділення *fs* на цільову СЧ *sys2*, перетворену з цілого числа на рядок; якщо після такої дії рядок залишку від ділення *od* пустий, йому присвоюється «0»; у цілочислову змінну *t1* потрапляє вміст залишку *od*, перетворений з рядка на число стандартною функцією *strtoint*; до рядка *r* додається символ, отриманий допоміжною функцією *ntoch* від *t1*; значення точності *prec* зменшується на 1.

Після отримання цілої частини у цільовій СЧ виконується ще кілька підготовчих дій до другого циклу: рядковій змінній майбутнього кінцевого

результату *r1* присвоюється результат роботи попереднього циклу *r*; рядок *r* очищується; точність конвертування та ділення *prec* відновлюється із змінної *pt*. Якщо перед основним циклом була встановлена наявність дробової частини, то починає роботу другий цикл «*while (prec>0) and (fp<>'0')*», в якому виконуються наступні інструкції:

1. у рядок *t* потрапляють всі цифри перед комою шляхом виконання команди *t:=copy(s,1,pos(comma,s)-1)*;
2. вхідний рядок *s*, ціла частина в якому перед початком даного циклу була замінена на «0», множиться на цільову СЧ *sys2* і результат зберігається в *s*;
3. формується рядок *ip* з цілої частини *s* вже описаним способом;
4. ціле число *pt* отримує значення *ip*, перетворене на integer;
5. у змінній *ip* відновлюється значення *pt*, отримане в результаті роботи команди *ntoch*;
6. вміст *ip* додається в кінець результату *r*;
7. якщо рядок *s* містить лише цілу частину (*pos(comma,s)=0*), *fp* отримує значення «0», що стає ознакою останньої ітерації даного циклу;
8. якщо рядок *ip*, доданий до результату, не дорівнює цілій частині *t*, отриманій на початку ітерації циклу, то ціла частина рядка *s* замінюється на «0»;
9. зменшується точність *prec* на 1, і при необхідності повторюються вказані операції.

Після завершення циклу отримана дробова частина *r* додається після коми рядка кінцевого результату *r1*, якщо другий цикл виконувався; здійснюється доробка результату шляхом видалення зайвих символів «0» на початку рядка та зайвої коми в кінці рядка (при необхідності). Отже, функція *convertfromdec* отримує значення *r1*.

Основна функція конвертера *numberconvert* служить для перетворення числа з будь-якої СЧ у десяткову. Усі арифметичні операції для перетворення здійснюються у вхідній СЧ *sys1*, тому змінна *sys* отримує значення *sys1*, її

значення дублюється також у цілому числі nst (для відновлення). Якщо вхідна СЧ *sys1* дорівнює 10, то дана функція завершується з результатом роботи вище описаної підпрограми *convertfromdec*. У іншому випадку виконуються наступні підготовчі операції: якщо перший символ вхідного рядка – «0» і в ньому є кома, то така ціла частина в ньому замінюється на «1»; рядкова змінна *r1*, яка буде використовуватись у циклі, отримує пусте значення; дробова частина вхідного операнду виділяється в рядок *ds* командою *copy(s,pos(comma,s)+1,length(s)-pos(comma,s))*, після чого вона видаляється із вхідного рядка *s*; цілочислова змінна *k* отримує довжину *s*, меншу на 1; рядок результату ініціалізується з символом «0»; запускається цикл «*while k>-1*» для перетворення цілої частини вхідного рядка у цільову СЧ, містить наступні дії:

1. у рядок *r1* потрапляє результат зведення початкової СЧ у степінь *k*;
2. символ *c* отримує цифру вхідного числа *s* з індексом *length(s)-k*, яка перетворюється функцією *chton* на ціле число *n*;
3. рядок *r1* оновлює своє значення шляхом множення самого себе на перетворене у рядок *n*;
4. до рядка кінцевого результату додається число *r1*: *t1:=plus(r1,t1)*;

Якщо рядок *ds* не порожній, то після виконання команди *k:=-length(ds)* починає роботу цикл конвертування дробової частини «*for i:=1 to length(ds)*», який виконує ті самі операції, що й попередній цикл, але початкова СЧ *sys1* зводиться у від'ємний степінь *i*.

Після одного або двох таких циклів (у залежності від числа) виконується перевірка умови: якщо цільова СЧ не 10, то початкова *sys1* отримує значення 10 і результат попередніх перетворень *t1* додатково оброблюється функцією *convertfromdec*, після чого при необхідності виконуються дії для корекції результату, зокрема, видаляються зайві нулі в цілій частині; значення початкової СЧ *sys1* відновлює значення, введене користувачем, із змінної nst.

Результати перетворень зображені на рисунку 5:

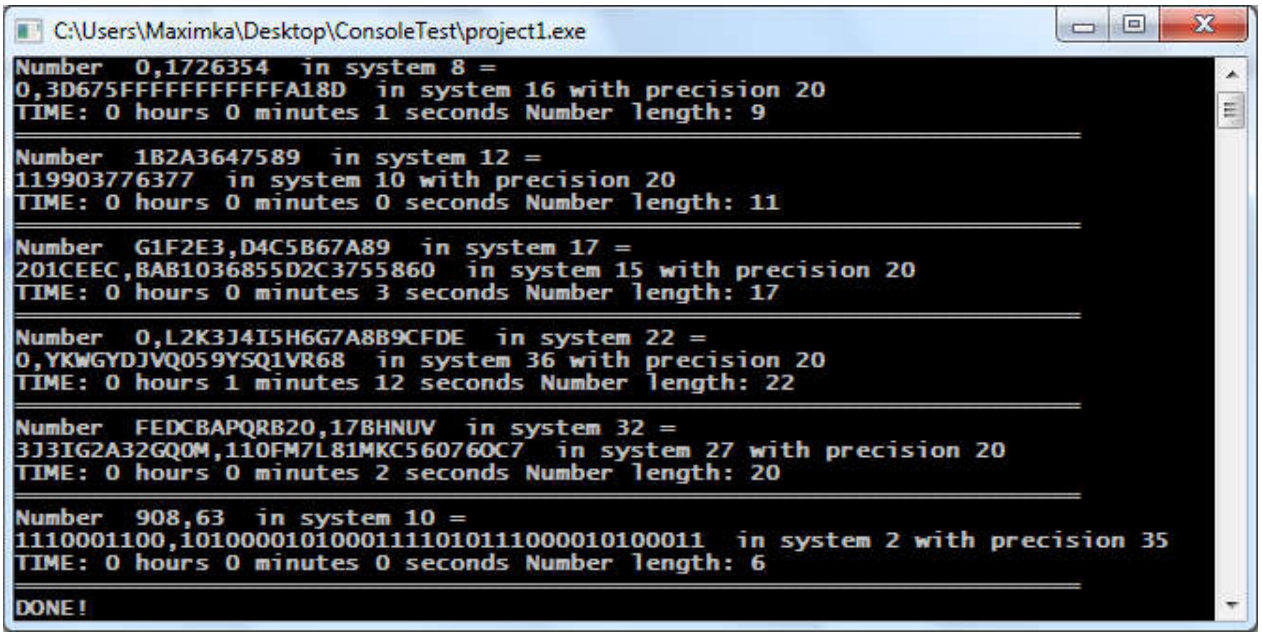

Рисунок 5 – Результати роботи конвертера систем числення

Усі надані результати майже повністю співпадають з результатами, наданими онлайн-сервісом numsys.ru: розбіжності починаються лише після 10-12 знаків після коми, що можна усунути збільшенням точності в інтерфейсі користувача. Також при тестуванні помічена недостатня швидкодія обробки чисел з довгою дробовою частиною: вона може тривати в деяких випадках до кількох хвилин, що спричинено недостатньою раціональністю реалізації алгоритму ділення.

3.4 Модуль розрахунку ваги металопрокату

Сукупність певних елементів інтерфейсу користувача віконної версії калькулятора дають змогу автоматизувати розрахунок ваги металопрокату за вказаними у відповідні поля даними. Даний модуль складається зі списків «Вид металла», «Тип и размер металлопроката», полів «Кол-во м», «Толщина», «Удельный вес (кг/п.м)», «Вес с учетом количества» та кнопки «Вычислить вес». В залежності від вибраного елементу списку «Вид металла» активуються усі інші поля та списки: якщо користувач вибрав «Листовой», поле «Толщина» стає активним, а два «випадні» списки «Тип и размер металлопроката» та «Удельный вес (кг/п.м)» вимикаються; в усіх інших випадках, при виборі «Сортовой» або

«Трубы», неактивним стає лише поле «Толщина».

Елементи списку «Тип и размер металлопроката» завантажуються з певних файлів в залежності від вибраного виду металу:

- якщо користувачем вибрано «Сортовой», перелік завантажується з текстового файлу *Names&Dimensions_sorted.txt*. При виборі елементів списку «Тип и размер металлопроката» у полі «Удельный вес (кг/п.м)» відображаються довідкові дані, які зчитуються з файлу *Weights_sorted.txt* циклом *for i:=0 to GostMetallType.ItemIndex do readln(f,s)*;
- якщо вибрано «Трубы», список найменувань завантажується з файлу *Names&Dimensions_pipe.txt*, а довідкові дані отримуються з файлу *Weights_pipe.txt* тим самим способом.

При натисканні на кнопку «Вычислить вес» виконуються такі дії:

1. система числення (поле *ns.text*) отримує значення 10, оскільки даний модуль працює лише в десятковій СЧ;
2. в залежності від активності певних полів здійснюється перевірка коректності введених даних функцією *checkfield* типу boolean, яка реалізована подібно до функції *check* у модулі калькулятора, але перевіряє числа лише у десятковій СЧ, при помилці видає повідомлення: «Введите корректные данные!», переставляє фокус на поле введення кількості (оскільки після натискання на кнопку він змінився), та повертає результат false (при правильності рядка результат – true);
3. при коректності введеної інформації формується арифметичний вираз в залежності від вибраного елементу списку «Вид металла»: якщо активним є елемент «Листовой», то вираз формується з чисел у полях «Кол-во м» та «Толщина» за формулою відповідно *quantity.text* *thickness.text*7,85*; у іншому випадку формула для розрахунку *quantity.text*unit_weight.text* («Кол-во м» * «Удельный вес (кг/п.м)»);
4. сформований рядковий вираз *s* потрапляє у поле для введення арифметичних операцій (*ae.text:=s*);

5. викликається процедура натискання на кнопку «=» і здійснюються обчислення із вказаними даними;
6. поле «Вес с учетом количества» отримує результат обчислень калькулятором і фокус повертається на поле «Кол-во м» (команда *quantity.SetFocus*).

Довідкова інформація, яка використовується даним модулем, може доповнюватись та редагуватись користувачем без зміни коду програми, хоча на даному етапі розвитку проекту така можливість реалізована не дуже зручно: найменування видів металопрокату та удільна вага кожного з них зберігаються у різних файлах. Тому користувач має пам’ятати, в якому саме файлі яка інформація знаходиться та при зміні даних слідкувати за відповідністю номерів рядків, що дуже зручно робити за допомогою розширених текстових реакторів, таких як Notepad++. На наш погляд, єдиним недоліком даного модуля є недостатня зручність зміни та доповнення довідкових даних.

3.5 Модулі розрахунків ваги, довжини діаметра та розривної сили каната або троса

Сукупність таких елементів інтерфейсу користувача як «випадні» списки «Размеры троса (мм)» та «Рассчитать», поля «Вес 1 м троса (кг)», «Количество метров» та «Вес троса (кг)», кнопка «Вычислить вес троса», а також відповідних їм підпрограм становить модуль розрахунку ваги троса. Він створений для надання можливості швидкого обчислення ваги троса за заданою у програмі формулою та з використанням довідкових даних лише у десятковій СЧ.

Список «Размеры троса (мм)» включає в себе лише два елементи: «по окружности» і «по диаметру». Елементи з даними завантажуються з файлу *rope_size&weight* в список, який знаходиться прямо біля нього, в залежності від вибраного пункту. Елементи даного списку розміщуються у вказаному текстовому файлі в такому вигляді: «по окружности» «по диаметру»/дані. Отже, формування відповідних переліків здійснюється такими двома циклами:

if type_of_size.Text='по окружности' then

```
while not eof(f) do begin
readln(f,s);
delete(s,pos(' ',s),length(s)-pos(' ',s)+1);
rope_s.Items.Add(s);
end  else
while not eof(f) do begin s:='';
readln(f,s);
delete(s,1,pos(' ',s));
delete(s,pos('/',s),length(s)-pos('/',s)+1);
rope_s.Items.Add(s);
end;
```

де *rope_s* – назва списку, *f* – змінна типу text (відкритий файл), *s* – рядок.

В залежності від вибраних елементів у поле «Вес 1 м троса (кг)» потрапляють відповідні довідкові дані, які в текстовому файлі знаходяться після символу «/».

Список «Рассчитать» містить у собі два пункти, вибір одного з яких змінює назви деяких елементів інтерфейсу: при виборі пункту «вес» поле, яке знаходиться поряд, отримує назву «Количество метров», кнопка перейменовується на «Вычислить вес троса», а останнє поле результату – на «Вес троса (кг)»; вибір елементу «длину» відповідно формує назви «Количество килограмм», «Вычислить длину троса» та «Длина троса (м)».

Натискання на кнопку під назвою «Вычислить вес троса» або «Вычислить длину троса» викликає попередню перевірку поля з назвами «Количество метров» або «Количество килограмм» вище згаданою функцією *checkfield* (у коді програми дане поле має лише одну назву *kgm*), встановлює значення поля СЧ на 10, формує арифметичний вираз таким чином: якщо користувач обчислює вагу, то вказана кількість метрів ділиться на вагу 1 м троса, в іншому випадку ці два поля перемножуються калькулятором, результат потрапляє в поле під назвою «Вес троса (кг)» або «Длина троса (м)». Якщо користувач обчислював вагу, до поля результату роботи даного модуля додається надпис «кг», якщо довжину – «м».

Проаналізувавши роботу цього невеликого модуля можна стверджувати, що він реалізований більш вдало, аніж попередній, оскільки має більш досконалу систему доповнення та редагування довідкових даних: достатньо відкрити відповідний текстовий файл і, пам'ятаючи про те що числами перед пробілом є окружність, після пробілу – довжина, а після символу «/» – вага (яка є спільною для «окружності» та «діаметра»), можна легко змінити та доповнити існуючі дані, при цьому не заплутуючись у різних файлах. Проте некоректно змінений текстовий файл може спричинити помилки при роботі даного модуля.

Останній «інженерний» модуль поєднує в собі можливості розрахунку розривного зусилля і діаметра каната. Він складається з двох списків «Режим работы каната» та «Тип каната в наличии», поля з довідковою інформацією «Коэффициент», поля «Усилие (кг)» для введення даних користувачем, двох полів для результату: «Разрывное усилие каната (кг)» і «Диаметр каната (мм)».

Список «Режим работы каната» завантажується з файлу *rope-cond.txt*, в якому довідкова інформація для відповідного режиму роботи знаходиться після символу «/». При виборі елементу даного списку довідкові дані потрапляють у поле «Коэффициент». Список «Тип каната в наличии» складається з трьох полів: «трехпрядный пеньковый», «стальной (6х36 - 216 проволок)» та «стальной (6х37 - 222 проволоки)», вибір одного з яких відповідно ініціалізує один з трьох файлів: *rope-diameter_hemp.txt* або *rope-diameter_iron216.txt* або *rope-diameter_iron222.txt* та частково повторює дії кнопки запуску обчислень для оновлення результату.

Кнопка «Вычислить разрывное усилие и диаметр каната» після всіх стандартних підготовчих дій формує простий арифметичний вираз за формулою «Усилие (кг)» * «Коэффициент», присвоює результат його обчислення полю «Разрывное усилие каната (кг)», при необхідності округлює його командою *inttostr(round(strtofloat(e3.text)))* (оскільки користувачу не потрібна дуже точна вага), після чого в залежності від вибраного елементу списку «Тип каната в наличии» відкриває для читання один з трьох вище згаданих файлів та здійснює пошук діаметра каната наступним циклом:

while not (eof(f)) do begin

```
readln(f,s);
if strtoint(bursting_effort.text)<=strtoint(copy(s,1,pos('/',s)-1))
then begin  delete(s,1,pos('/',s));
rope_d.text:=s; closefile(f); exit; end; end;
```

в якому *f* – файл типу text, *s* – зчитаний рядок, а *bursting_effort* – поле «Разрывное усилие каната (кг)». Якщо у циклі не відбувся вихід з підпрограми, процедура встановлює полю «Диаметр каната (мм)» значення «Большое усилие!». Той самий цикл виконується при виборі іншого елементу списку «Тип каната в наличии» за умовою якщо поле «Разрывное усилие каната (кг)» не є пустим (тобто кнопка для відповідного розрахунку вже була натиснута користувачем після введення коректних даних).

ВИСНОВКИ

Метою дипломного проекту була розробка програмно-апаратного комплексу підтримки інженерних розрахунків для машинобудівного підприємства з такими можливостями:

- здійснення обчислень у всіх СЧ над цілими та дробовими знаковими числами великої довжини;
- перетворення чисел з однієї СЧ в іншу з заданою точністю;
- автоматизація деяких інженерних розрахунків із використанням достовірних довідкових даних з можливістю їх зміни;
- відображення здійснених операцій та часу їх виконання.

Розроблена програма була успішно випробувана при вирішенні реальних виробничих задач. Нами не було виявлено розбіжностей результатів її роботи з іншими аналогічними, але менш функціональними системами. Вона може вважатись саме програмно-апаратним комплексом, оскільки частково імітує роботу деяких апаратних пристроїв, таких як кишенькові калькулятори, і, безсумнівно, не може працювати без належної як апаратної так і програмної підтримки.

У ході виконання дипломного проекту було проаналізовано більшість ПЗ, яке використовується на сучасних вітчизняних машинобудівних підприємствах, докладно описані «прямі» аналоги розробленого програмного комплексу, висвітлені їх недоліки. Нами докладно прокоментовані алгоритми розробленої програми, висвітлені вимоги для її функціонування та підтримки іншими спеціалістами.

Програмний комплекс планується до введення в експлуатацію на підприємстві, що підтверджується довідкою про впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. M. van Canneyt, M. Gartner, S.Heinig, F.Monteiro de Cavalho, I.Ouedraogo. Lazarus, the Complete Guide. — Blaise Pascal Magazine, 2011. — 735 с.
2. Mattias Gaertner Lazarus for Cross-Platform Development (англ.) // Linux Journal. — Belltown Media, Inc., 2009. — Fasc. 185.
3. Алексеев Е. Р., Чеснокова О. В., Кучер Т. В. Free Pascal и Lazarus: Учебник по программированию. — М.: Альт Линукс, ДМК Пресс, 2010. — 440 с. — (Библиотека ALT Linux).
4. Гашков С. Б. Системы счисления и их применение. — М.: МЦНМО, 2004. — (Библиотека «Математическое просвещение»).
5. Гуриков С.Р. Программирование в среде Lazarus для школьников и студентов. — -М.: ФОРУМ, 2016. — 336 с.
6. Иан Соммервилл. Инженерия программного обеспечения / Пер. с англ. — 6-е издание. — М.: Вильямс, 2002. — 624 с.
7. Микушин А. В. Системы счисления. Курс лекций «Цифровые устройства и микропроцессоры».
8. Нил Дж. Рубенкинг. Язык программирования Delphi для «чайников». Введение в Borland Delphi 2006 = Delphi for Dummies. — М.: Диалектика, 2007. — 336 с.
9. Роберт У. Себеста. Основные концепции языков программирования / Пер. с англ. — 5-е изд. — М.: Вильямс, 2001. — 672 с.
10. Фомин С. В. Системы счисления. — М.: Наука, 1987. — 48 с. — (Популярные лекции по математике).
11. Яглом И. Системы счисления // Квант. — 1970. — № 6. — С. 2-10.
12. Иан Грэхем. Объектно-ориентированные методы. Принципы и практика / Пер. с англ. — 3-е изд. — М.: Вильямс, 2004. — 880 с.