МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

В. В. Третяк

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕХНОЛОГІЧНИХ РОЗРАХУНКІВ В ОБ'ЄКТНО-ОРІЄНТОВАНОМУ СЕРЕДОВИЩІ

Навчальний посібник

Харків «ХАІ» 2021

УДК 004.896(075.8)
Т79

Рецензенти: канд. техн. наук, доц. В. М. Дьоміна,
канд. техн. наук, доц. М. В. Репетенко

Третяк, В. В.

Т79 Розроблення програмного забезпечення для технологічних розрахунків в об'єктно-орієнтованому середовищі [Текст] : навч. посіб. / В. В. Третяк. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2021. – 80 с.

ISBN 978-966-662-846-9

Описано структуру, організацію і методи розроблення програмного забезпечення для виконання технологічних розрахунків в об'єктно-орієнтованому середовищі. Подано приклади розроблення програмного комплексу для виконання розрахунків режимів різання при проектуванні технологічних процесів деталі авіаційного двигуна. Наведено первинні тексти програм.

Для студентів, що вивчають курси «Технологія виробництва двигунів та енергетичних установок ЛА», «САПР ТП», «Технологія об'єктно-орієнтованого програмування», «Фізико-хімічні основи технологічних процесів», «Методи і параметри формоутворювальної обробки».

Іл. 20. Табл. 34. Бібліогр.: 12 назв

УДК 004.896(075.8)

ISBN 978-966-662-846-9

ВСТУП

Зміст інженерної діяльності пов'язаний із забезпеченням реалізації життєвого циклу (ЖЦ) складних виробів.

Відповідно до стандарту ISO 9004 життєвий цикл виробу складається з таких етапів:

1. Маркетинг, пошук і вивчення ринку.
2. Проектування і конструювання виробу.
3. Матеріально-технічне постачання.
4. Технологічна підготовка виробництва.
5. Виробництво, контроль і проведення випробувань.
6. Упаковування і зберігання.
7. Реалізація і/або розподіл продукції.
8. Монтаж і експлуатація.
9. Технічна допомога в обслуговуванні.
10. Утилізація.

Ці етапи можуть частково доповнювати один одного.

Також ЖЦ одного покоління виробів може переходити в ЖЦ наступного покоління і т. д. ЖЦ авіатехніки характеризується великою протяжністю окремих етапів.

Наприклад, створення літака або авіадвигуна триває лише до 5–10 років.

Скорочення виготовлення виробів є можливим завдяки розробленню САПР систем, які використовували б сучасні методи програмування.

В останні десятиріччя в наукомістких і високотехнологічних галузях промисловості розвинутих країн активно впроваджуються роботи зі скорочення строків розроблення виробів унаслідок математичного моделювання процесів і скорочення експериментальних досліджень з використанням баз знань.

Розробник баз знань повинен не тільки володіти своєю спеціальністю, але й мати навички розроблення своїх програмних модулів для її використання у роботі технолога.

Цю роботу легко можна виконувати в об’єктно-орієнтованому середовищі DELPHI.

Ця система є класичною для навчання і розуміння.

У навчальному посібнику наведено загальний опис, правила і приклади розроблення програм в об’єктно-орієнованому середовищі, які могли б бути використані технологами при розробленні нових технологічних процесів, подано приклади розроблення програмних комплексів, а також первинні тексти програм.

1. ОСОБЛИВОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Мова, яка реалізує концепцію об'єктно-орієнтованого програмування (ООП), означає, що функціональність додатка визначається набором взаємозв'язаних завдань, кожне з яких стає самостійним об'єктом. У об'єкта є властивості (тобто характеристики або атрибути), методи, визначальні його поведінка і події, на які він реагує. Одним з найважливіших понять ООП є клас. Клас є подальшим розвитком концепції типу і об'єднує в собі завдання не тільки структури і розміру змінних, але й виконуваних над ними операцій. Об'єкти в програмі завжди є екземплярами того або іншого класу (аналогічно змінним певного типу).

1.1. Основні концепції ООП

До основних концепцій ООП належать: інкапсуляція, спадкоємство, і поліморфізм.

Інкапсуляція є об'єднанням даних і методів, що їх обробляють (підпрограм) усередині класу (об'єкта). Це означає, що в класі інкапсулюються (об'єднуються і поміщаються всередину) поля, властивості і методи. При цьому клас чекає певної функціональності, забезпечуючи, наприклад, повний набір засобів для створення програми підтримки деякого елемента інтерфейсу (вікна Windows редактора і т. п.) або прикладної обробки.

Спадкоємство — це процес породження нових об'єктів-нащадків від існуючих об'єктів-батьків, при цьому нащадок успадковує від батька всі його поля, властивості і методи. Успадковані поля, властивості і методи можна використовувати в незмінному вигляді або перевизначати (модифікувати). Просто спадкоємство великого значення не має, тому до об'єкта-нащадка добавляються нові елементи, що визначають його особливість і функціональність.

Видалити будь-які елементи батька в нащадку не можна. У свою чергу, від нового об'єкта можна породити наступний об'єкт, у результаті утворюється дерево об'єктів (ієрархія класів). У корені цього дерева знаходиться базовий клас TObject, який реалізує найбільш загальні для всіх класів елементи, наприклад дії зі створення і видалення об'єкта. Чим далі той або інший клас відстоїть у дереві від базового класу, тим більшу специфічність він має.

Приклад нового об'явлення класу:TAnyClass = class (TParentClass);

```
// Додавання до класу TParentClass
// нових  і перевизначення існуючих елементів
...
end;
```

Поліморфізм полягає у тому, що методи різних класів можуть мати однакові імена, але різний зміст. Це досягається визначенням батьківського методу в класі-нащадку. В результаті батько і нащадок поводяться по-різному.

При цьому звернення до однойменних методів різних об'єктів виконується аналогічно.

1.2. Класи і об'єкти

У мові Object Pascal класи — це спеціальні типи даних, що використовується для опису об'єктів. Відповідно, об'єкт, що має тип якого-небудь класу, і є екземпляром (instance) цього класу або змінними цього типу.

Клас є особливим типом запису, має у своєму складі такі елементи (члени), як поля, властивості і методи. Поля класу аналогічні полям запису і призначені для зберігання інформації про об'єкт. Методами називаються процедури і функції, призначені для оброблення полів.

Властивості займають проміжне положення між полями і методами. З одного боку, властивості можна використовувати як поля, наприклад присвоюючи їм значення за допомогою інструкції присвоєння, з іншого, усередині класу доступ до значень властивостей виконують методи класу.

Опис класу має таку структуру:

```
type <Им’я класу> = class(<Им'я класу-батьків>)
private
<Приватні описи>;
protected
<Захищені описи>;
public
<Загальнодоступні описи>;
published
<Опубліковані описи>;
end;
```

У наведеній структурі описами є оголошення властивостей, методів і подій.

Приклад опису класу:

```
type
TColorCircle = class(TCircle)
Fleft,  Ftop,
Fright,
FBottom: Integer;
Color: TColor;
end;
```

Тут клас TColorCircle створюється на основі батьківського класу TCircle. порівняно з батьківським новий клас додатково містить чотири поля типу Integer і одне поле типу TColor.

Якщо як батьківський використовують клас Tobject, який є базовим класом для всіх класів, то його ім'я після слова class можна не указувати. Тоді перший рядок опису виглядатиме так: type TNewClass = class.

Для різних елементів класу можна встановлювати різні права доступу (видимости), для чого в описі класу використовуються окремі розділи, позначені спеціальними специфікаторами видимості.

Розділи **private** і **protected** містять захищені описи, які доступні усередині модуля, в якому вони знаходяться.

Описи з розділу protected, крім того, доступні для породжених класів за межами названого модуля.

Розділ **public** містить загальнодоступні описи, які видимі в будь-якому місці програми, де доступний сам клас.

Розділ **published** містить опубліковані описи, які на додаток до загальнодоступних описів породжують динамічну (тобто під час виконання програми) інформацію про тип (Run-Time Type Information, RTTI).

За цією інформацією при виконанні додатка проводиться перевірка на належність елементів об'єкта до того або іншого класу. Одним з призначень розділу **published** є забезечення доступу до властивостей об'єктів при конструюванні додатків.

В інспекторі об'єктів видно ті властивості, які є опублікованими. Якщо специфікатор **published** не вказано, то він виконується за умовчанням, тому будь-які описи, розташовані за рядком із зазначенням імені класу, вважаються опублікованими.

Об'єкти як екземпляри класу оголошуються у програмі в розділі **var** як звичайні змінні.

Наприклад:

```
var
CCircle1: TColorCircle;
CircleA: TCircle;
```

Як і у разі записів, для звернення до конкретного елемента об'єкта (поля, властивості або методу) указуються ім'я об'єкта та ім'я елемента, розділені крапкою, тобто ім'я елемента є складовим.

Приклад звернення до полів об'єкта:

```
var
CCircle1: TColorCircle;
begin
...
CCircle1.FLeft:=5; CCircle1.FTop:=1;
... end;
```

2. СЕРЕДОВИЩЕ DELPHI

Прикладні програми, або додатки, Delphi створюються в інтегрованому середовищі розроблення (IDE — Integrated Development Environment). Вони призначені для користувача, а інтерфейс цього середовища — для організації взаємодії з програмістом і містить декілька вікон з різними елементами керування. За допомогою засобів інтегрованого середовища розробнику зручно проектувати інтерфейсну частину додатка, а також писати програмний код і зв'язувати його з елементами керування. В інтегрованому середовищі розроблення проходять всі етапи створення додатка та його налагоджування. Інтегроване середовище розроблення Delphi є багатовіконною системою. Вид інтегрованого середовища розроблення (призначений для користувача інтерфейс) може відрізнятися залежно від настройок. Після завантаження інтерфейс Delphi має вигляд, показаний на рис. 2.1, і містить шість вікон:

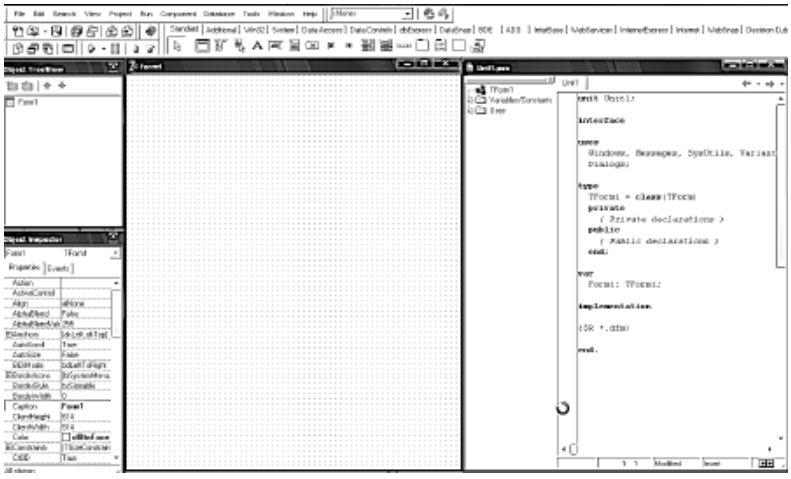

Рис. 2.1. Сторінки середовища Delphi

— головне вікно (Delphi — Project1);
— вікно Оглядача дерева об'єктів (Object TreeView);
— вікно Інспектора об'єктів (Object Inspector);
— вікно Форми або Конструктора форми (Form1);
— вікно Редактора коду (Unit1.pas).
— вікно Провідника коду (Exploring Unit1.pas).

Останні два вікна знаходяться позаду вікна Форми, причому вікно Провідника коду пристиковано зліва до вікна Редактора коду, тому обидва ці вікна мають загальний заголовок Unit1.pas. На екрані окрім зазначених вікон можуть бути й інші вікна, що відображаються при виклику відповідних засобів, наприклад вікно Редактора зображень (Image Editor). Вікна Delphi

можна переміщувати, змінювати їх розміри і видаляти з екрана (окрім головного вікна), а також зістиковувати між собою. Незважаючи на наявність багатьох вікон Delphi є однодокументарним середовищем і дозволяє одночасно працювати тільки з одним додатком (проектом додатка). Назва проекту додатка виводиться у рядку заголовка головного вікна у верхній частині екрана. При згортанні головного вікна згортається весь інтерфейс Delphi і всі відкриті вікна, а при закритті головного вікна робота з Delphi припиняється.

Головне вікно Delphi містить:

— головне меню;
— панелі інструментів;
— палітру компонентів.

Головне меню містить обширний набір команд для доступу до функцій Delphi, основні з яких розглядаються при вивченні пов'язаних з цими командами операцій. Панелі інструментів знаходяться під головним меню у лівій частині головного вікна і мають п'ятнадцять кнопок для виклику команд, що найбільш часто використовуються з головного меню, наприклад File | Open (Файл | Відкрити) або Run | Run (Виконання | Виконати). Викликати команди головного меню можна також за допомогою комбінацій клавіш, які указуються праворуч від назви відповідної команди. Наприклад, команду Run | Run можна викликати за допомогою клавіші <F9>, а команду View | Units (Перегляд | Модулі) — за допомогою комбінації клавіш <Ctrl>+<F12>. Всього маємо шість панелей інструментів:

— Standard (Стандартна);
— View (Перегляд);
— Debug (Налаштування);
— Custom (Користувач);
— Desktop (Робочий стіл);
— Internet (Інтернет).

Відображенням панелей інструментів і настроюванням кнопок на них можна керувати. Ці дії виконуються за допомогою контекстного меню панелей інструментів, яке визивається клацанням правої кнопки миші при розміщенні покажчика в області панелей інструментів або головного меню. За допомогою контекстного меню можна також керувати видимістю Палітри компонентів (Component Palette).

Більш широкі можливості щодо настроювання панелей інструментів і головного меню надає показане діалогове вікно Customize (Індивідуальне настроювання), що викликається однойменною командою контекстного меню панелей інструментів.

З його допомогою можна приховати або відобразити необхідну панель инструментів, змінити склад кнопок на ній, а також вибрати режим відображення спливаючих підказів для кнопок.

3. МОВА ПРОГРАМУВАННЯ У СЕРЕДОВИЩІ DELPHI

Починаючи з версії 7, у середовищі Delphi для розроблення додатків використовується мова програмування Delphi, основою якої є мова Object Pascal (об'єктно-орієнтоване розширення стандартної мови Pascal).

Програмування на мові Delphi призначено для роботи в інтегрованому середовищі розроблення додатків (IDE).

При цьому система накладає ряд обмежень, які виходять за межі специфікації мови Object Pascal. Зокрема, посилюються угоди про найменування файлів і програм, яких не обов'язково дотримуватися за межами IDE.

Система мови Delphi забезпечує можливість візуального програмування на ній за допомогою бібліотеки візуальних компонентів VCL. У середовищі Delphi можна використовувати бібліотеку Borland CLX (Component Library for Cross-Platform), що являє собою міжплатформний варіант бібліотеки VCL, призначений для розроблення додатків під Windows і Linux.

У цьому розділі розглянемо основні засоби і прийоми програмування на мові Delphi.

3.1. Основні поняття

Розглянемо алфавіт, словник, структуру програми, коментарі, типи даних, інструкції і директиви компілятора, що використовується в мові Delphi.

3.2. Алфавіт

Алфавіт мови Delphi містить такі символи:

— 53 букви — великіі (A...Z) і малі (a...z) букви латинського алфавіту, а також:
— символ підкреслення (_);
— 10 цифр (0...9);
— 23 спеціальні символи (+ - * / . , : ; = > < ' ( ) { } [ ] # $ ^ @ і пропуск).

Комбінації спеціальних символів утворюють такі складові символи:

— := (присвоєння);
— <> (не дорівнює);
— .. (діапазон значень);
— <= (менше або дорівнює);
— >= (більше або дорівнює);
— (* і *) (альтернатива фігурним дужкам { і });
— (. і .) (альтернатива квадратним дужкам [і]).

3.3. Словник мови

Неподільні послідовності знаків алфавіту утворюють слова, які відокремлюються один від одного роздільниками і мають певне значення в програмі.

Роздільниками можуть бути пропуск, символ кінця рядка, коментар, інші спеціальні символи та їх комбінації.

Слова підрозділяються:

— на ключові слова;

— стандартні ідентифікатори;

— ідентифікатори, призначені для користувача.

Ключові (зарезервовані) слова є складовою частиною мови, мають фіксоване написання і однозначно певне значення, змінити який програміст не може.

Наприклад, ключовими є слова: Label, Unit, Goto, Begin та ін.

У Редакторі коду ключові слова виділяються напівжирним шрифтом. Стандартні ідентифікатори позначають наперед визначені розробниками конструкції мови:

— типи даних;

— константи;

— процедури і функції.

На відміну від ключових слів будь-який із стандартних ідентифікаторів можна перевизначити, але, оскільки це може призвести до помилок, стандартні ідентифікатори все ж таки краще використовувати без будь-яких змін. Прикладами стандартних идентифікаторів є слова Sin, Pi, Real. Призначені для користувача ідентифікатори використовують для позначення імен міток, констант змінних, процедур, функцій і типів даних. Ці імена задаються програмістом і мають відповідати таким правилам:

— ідентифікатор складається з букв і цифр;

— ідентифікатор завжди починається тільки з букви, виключенням є мітки, якими можуть бути цілі числа без знака в діапазоні 0...9999;

— в ідентифікаторі можна використовувати як великі, так і малі букви, компілятор інтерпретує їх однаково;

— між двома ідентифікаторами у програмі має бути принаймні один роздільник.

3.4. Структура програми

Початковий текст програми подається у вигляді послідовності рядків, при цьому текст рядка може починатися з будь-якої позиції. Структурно програма складається з заголовка і блока. Заголовок знаходиться на початку програми і має вигляд:

Program <Ім’я програми>;

Блок складається з двох частин: описової та виконавчої.

В описовій частині міститься опис елементів програми, а у виконавчій указуються дії з різними елементами програми, які дозволяють отримати результат, що потребується. У загальному випадку описова частина складається з наступних розділів:

— підключення модулів;
— об’явлення міток;
— об’явлення констант;
— описи типів даних;
— об’явлення змінних;
— описи процедур і функцій.

У кінці кожного із зазначених розділів ставиться крапка з комою.

Розглянемо відмінність між термінами "об’явлення" і "опис".

Суть її полягає в тому, що об’явлення деякого об'єкта у програмі припускає виділення основної пам'яті для його розміщення. Опис деякої конструкції у програмі, на відміну від об’явлення, не потребує виділення пам'яті. Структуру програми у загальному випадку можна подати так:

```
Program <Ім’я програми>;
  Uses <Список модулей>;
 Label <Список міток>;
  Const <Список констант>;
 Type <Опис типів>;
Var <Об’явлення змінних>;
     <Опис процедур>;
     <Опис функцій>;
        Begin
     <інструкції>;
        End.
```

У структурі конкретної програми будь-якого з розділів опису і об’явлення може не бути. Розділи описів і об’явлень, окрім розділу підключення модулів, який розташовується відразу після заголовка програми, можуть зустрічатися у програмі декілька разів і йти в довільному порядку. Розглянемо докладніше окремі розділи програми.

Розділ підключення модулів складається із зарезервованого слова Uses і списку імен стандартних і призначених для користувача бібліотечних модулів, що підключаються.

Формат цього розділу:

Uses <Ім’я1> <Ім’я2> ... <Ім’яN>;

Наприклад:

Uses Crt, Dos, MyLib;

Розділ об’явлення міток починається із зарезервованого слова Label, за яким ідуть імена міток, розділені комами.

Формат цього розділу:

Label <ім’я1> <ім’я2> ... <ім’яN>;

У програмі цей розділ матиме такий вигляд:

Label mitka1, mitka2, 10, 567;

У розділі об’явлення констант ідентифікаторам констант присвоюються їх значення.

Розділ починається з ключового слова Const, за яким ідуть ряд конструкцій, які присвоюють константам свої значення. Ці конструкції є ім'ям константи і виразом, значення якого присвоюється константі. Ім'я константи відділено від виразу знаком рівності, в кінці конструкції ставиться крапка з комою.

Формат цього розділу:

Const <ідентифікатор1> = <Вираз 1>;
...
<ідентифікаторN> = <Вираз N>;

Приклад об’явлення констант:

Const st1 = 'WORD'; ch = '5'; n34 = 45.8;

Компілятор автоматично розпізнає тип константи на підставі типу виразу. В Delphi є багато констант, які можна використовувати без попереднього об’явлення, наприклад Nil, True і MaxInt. У розділі опису типів описуються призначені для користувача типи даних. Цей розділ не є обов'язковим, і типи можуть бути описані неявно у розділі об’явлення змінних. Розділ опису типів починається з ключового слова Type, за яким розташовуються імена типів і їх опису, розділені знаком рівності.

У кінці опису ставиться крапка з комою.

3.5. Формат розділу типів

Type <Ім’я типу1> = <Опис типу1>;
...
<Ім’я типуN> = <Опис типуN>;

Наприклад:

Type char2 = ('a'..'z');
massiv = array[1..100] real;
month = 1..12;

В Delphi є багато стандартних типів, що не потребують попереднього опису: Real, Integer, Char, Boolean та ін.

Кожна змінна програми має бути об’явлена.

Об’явлення обов'язково передує використовуванню змінних.

Розділ об’явлення змінних починається з ключового слова Var, після якого через коми перелічуються імена змінних і через двокрапку їх тип (для кожного типу — свій список змінних, роздільник — крапка з комою). Формат розділу:

Var <ідентифікатори 1> : <тип1>;
...
<ідентифікатори N> : <типN>;

Наприклад:
Var a, bhg, u7: real;
simvol: char;
n1,n2: integer;

Об’явлення змінних забезпечує виділення пам'яті для розміщення змінних відповідно до їх типів, але не для присвоєння їм початкових значень. Програміст повинен самостійно задати потрібні початкові значення змінним перед їх використанням.

Підпрограмою називають логічно закінчену і спеціальним чином оформлену частину програми, яка може викликатися для виконання з інших крапок програми необмежену кількість разів.

У мові Delphi підпрограми поділяють на два типи: процедури і функції. Кожна підпрограма є блоком і має бути визначена у розділі опису процедур і функцій.

Опис процедур і функцій розглядається далі.

3.6. Формат розділу інструкцій

Розділ інструкцій починається з ключового слова Begin, після якого йдуть инструкції мови, розділені крапкою з комою. Завершує цей розділ ключове слово End, після якого указується крапка.

Формат розділу має такий вигляд:

Begin
<інструкція1>;
...
<інструкціяN>;
End.

Тут можуть використовуватися будь-які інструкції мови, наприклад інструкція присвоєння або умовна інструкція.

3.7. Коментарі

Коментар є текстом пояснення, який можна записувати у будь-якому місці програми, де дозволено пропуск.

Текст коментаря обмежений символами (* і *) (або їх еквівалентами { і }) і може містити будь-які символи мови, у тому числі українські і російські букви.

Коментар, обмежений вказаними символами, може займати декілька рядків програми. Однорядковий коментар містить подвійний слеш (//) на початку рядка:

Приклади коментарів:
(* Однорядковий коментар *)
// Інший однорядковий коментар
(* Початок багаторядкового коментаря
...
Закінчення багаторядкового коментаря *)

Коментар ігнорується компілятором і не робить ніякого впливу на виконування програми. За допомогою коментарів можна виключати будь-які інструкції програми в процесі її налаштування, наприклад так:

```
sum := 0;
for n := 1 to 100 do begin
read(x);
// if x < 0 then x := 0;
sum := sum + x;
end;
```

Тут умовна інструкція в тілі циклу оформлена як коментар і не буде виконуватися.

3.8. Типи даних

Оброблювані у програмі дані підрозділяються на змінні, константи і літерали. Константи є даними, значення яких встановлені у розділі об’явлення констант і не змінюються під час виконання програми.

Змінні об’являються в розділі об’явлення змінних, проте на відміну від констант одержують свої значення вже в процесі виконання програми, причому допускаються змінення цих значень. До констант і змінних можна звертатися по іменах.

Літерал не має ідентифікатора і подається в тексті програми безпоседньо значенням, тому літерали також називають просто значеннями.

Кожний елемент даних належить до певного типу, при цьому тип змінної указується при її описі, а тип константи і літерала розпізнається компілятором автоматично залежно від вказаного значення.

Тип визначає безліч значень, які можуть мати елементи даних і сукупність допустимих над ними операцій.

Наприклад, значення 34 і 67 належать до цілого типу, їх, відповідно, можна множити, складати, ділити і виконувати інші арифметичні операції, а значення 'abcd' і 'sdfh123' належать до строкового типу, і їх можна сполучати (складати), але не можна ділити або віднімати.

Типи даних можна поділити на такі групи

— прості;

— структурні;

— показникові;

— процедурні;

— варіантні.

У свою чергу, прості й структурні типи — це теж групи, до складу яких входять інші типи, наприклад цілочислові або масиви.

Важливе значення має поняття сумісності типів, яке означає, що типи дорівнюють один одному або один з них може бути автоматично перетворений на інший.

Сумісними, наприклад є дійсний і цілочислові типи, оскільки ціле число автоматично перетвориться на дійсне (але не навпаки).

3.9. Типи інструкцій

Інструкції є закінченими пропозиціями мови, які виконують деякі дії над даними. Інструкції Delphi можна поділити на дві групи:

— прості;

— структуровані.

Наприклад, до простих інструкцій належить інструкція присвоєння, а до структурнних — умовна інструкція і інструкція циклу. Інструкції розділяються між собою крапкою з комою. Наявність між інструкціями декількох крапок з комою не є помилкою, оскільки вони позначають порожні інструкції. Проте слід мати на увазі, що зайва крапка з комою у розділі описів і оголошень уже буде синтаксичною помилкою.

Крапка з комою може не ставитися після слова begin і перед словом end, оскільки вони розглядаються як операторні дужки, а не як інструкції.

В умовних інструкціях та інструкціях вибору крапка з комою (;) не ставиться після слова then і перед словом else.

Зазначимо, що в інструкції циклу з параметром наявність крапки з комою відразу після слова do не є синтаксичною помилкою, але у цьому випадку тіло циклу міститиме тільки порожню інструкцію.

4. ХАРАКТЕРИСТИКА ПРОЕКТУ

У цьому розділі розглядаються склад проекту, файл проекту, файли форми, файли модулів, файл ресурсів і параметри проекту.

4.1. Склад проекту

Створюваний у середовищі Delphi додаток складається з декількох елементів, об’єднаних у проект. Зв'язок між файлами проекту має форму, подану на рис. 4.1.

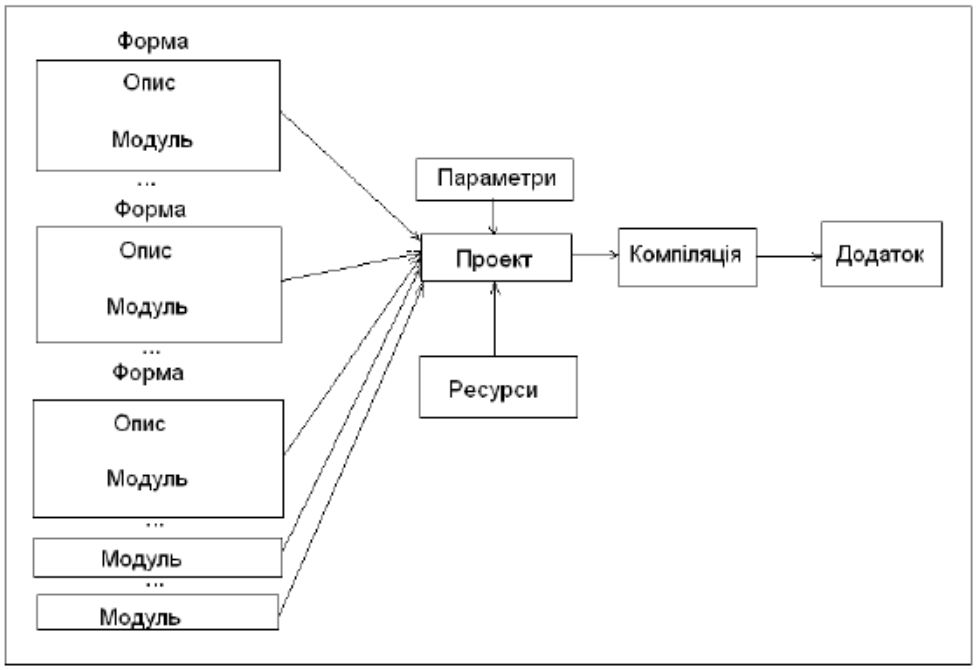

Рис. 4.1. Зв'язок між файлами проекту

До складу проекту входять такі елементи (у дужках зазначається розширення імен файлів):

— код проекту (dpr);
— описи форм (dfm);
— модулі і модулі форм (pas);
— параметри проекту для Windows (dof), для Linux (kof);
— параметри середовища (cfg);
— опис ресурсів (res).

Окрім наведених файлів автоматично можуть створюватися й інші файли, наприклад резервні копії файлів: ~dp — для файлів з розширенням dpr; ~pa — для файлів з розширенням pas.

При запуску Delphi автоматично створюється новий проект з ім'ям Project1, що відображається в заголовку головного вікна Delphi. Цей проект має у своєму складі одну форму Form1, назву якої видно у вікні Форми.

Розробник може змінити пропоноване за замовчуванням ім'я проекту, а також визначити параметри середовища таким чином, що після завантаження Delphi автоматично завантажуватиметься додаток, розроблення якого виконувалося востаннє. Зазвичай файли проекту розташовуються в одному каталозі. Оскільки навіть порівняно простий проект включає достатньо багато файлів, а при додаванні до проекту нових форм кількість цих файлів збільшується, для кожного нового проекту доцільно створювати окремий каталог, де і зберігати всі файли проекту.

4.2. Файл проекту

Файл проекту є центральним файлом проекту і його програмою. Для додатка, що має у складі одну форму, файл проекту має такий вигляд:

```
program Project1;
uses Forms Unit1 in 'Unit1.pas' {Form1}; {$R *.res}
begin
Application.Initialize;
Application.CreateForm(TForm1, Form1);
Application.Run; end.
```

Ім'я проекту (програми) збігається з ім'ям файла проекту і вказується при запису цього файла, первинне це ім'я — Project1. Таке ж ім'я мають файли ресурсів і параметрів проекту, при перейменуванні файла проекту ці файли перейменовуються автоматично.

Складання всього проекту виконується при компіляції файла проекту. При цьому ім'я розроблюваного додатка (exe-файл) або динамічно завантажуваної бібліотеки (dll-файл) збігається з назвою файла проекту. Надалі слід мати на увазі, що створюється додаток, а не динамічно завантажувана бібліотека.

У розділі uses указується ім'я модуля Forms, що підключається, який є обов’язковим для всіх додатків, що мають у своєму складі форми. Крім того, у розділі uses перелічуються модулі, що підключаються.

Директива $R підключає до проекту файл ресурсів, ім'я якого за замовчуванням збігається з ім'ям файла проекту. Тому замість імені файла ресурсу вказано символ *. Окрім цього файла розробник може підключити до проекту й інші ресурси, самостійно додавши директиви $R і вказавши в них відповідні імена файлів ресурсів. Ресурси, вказані в директиві $R, підключаються до проекту при його компіляції і складанні.

Тому бажано підключати у такий спосіб тільки порівняно невеликі за значенням витрати пам'яті ресурси, наприклад значки або курсори. Великі растрові зображення краще підключати динамічно, використовуючи

відповідні методи, наприклад LoadFromFile. Програма проекту містить всього три інструкції, які виконують ініціалізацію додатка, створення форми Form1 і запуск додатка. При виконанні розробником яких-небудь операцій з проектом Delphi формує код файла проекту автоматично. Наприклад, при додаванні нової форми у файл проекту додаються два рядки коду, що належать до цієї форми, а при виключенні форми з проекту ці рядки автоматично виключаються. За необхідності програміст може вносити змінення у файл проекту самостійно, проте подібні дії можуть зруйнувати цілісність проекту і тому зазвичай виконуються тільки досвідченими програмістами. Зазначимо, що деякі операції, наприклад створення обробника події для об'єкта Application, системою Delphi автоматично не виконуються і потребують самостійного кодування у файлі проекту. Відображення коду файла проекту у вікні Редактора коду задається командою Project | View Source (Проект | Проглядання джерела).

4.3. Файли форми

Для кожної форми у складі проекту автоматично створюються файл опису форми (розширення dfm) і файл модуля форми (розширення pas).

Файл опису форми є ресурсом Delphi і містить характеристики форми і її компонентів. Розробник зазвичай керує цим файлом через вікна Форми і Інспектора об'єктів. При конструюванні форми у файл опису автоматично вносяться відповідні змінення.

Файл опису форми є ресурсом Delphi, оскільки він розроблений саме для цього середовища і інтерпретується ним при створенні форми додатка. Зміст файла опису форми визначає її вигляд. За необхідності можна відобразити цей файл на екрані в текстовому вигляді, що виконується командою View as Text (Переглянути як текст) контекстного меню форми. При цьому вікно Форми пропадає з екрана, а замість файла опису форми відкривається у вікні Редактор кода і є доступним для перегляду і редагування.

4.4. Файли модулів

Окрім модулів у складі форм при програмуванні можна використовувати і окремі модулі, не пов'язані з якою-небудь формою. Вони оформляються за звичайними правилами мови Object Pascal і зберігаються в окремих файлах. Для підключення модуля його ім'я указується у розділі uses того модуля або проекту, який використовує засоби цього модуля. В окремому модулі можна (навіть корисно) розміщувати процедури, функції, константи і змінні, загальні для декількох модулів проекту.

5. ІНСТРУКЦІЇ МОВИ

5.1. Прості інструкції

Простими називаються інструкції, що не містять інших інструкцій. До них належать:

— інструкція присвоєння;
— інструкція переходу;
— порожня інструкція;
— інструкція виклику процедури.

5.2. Інструкція присвоєння

Інструкція присвоєння є основною інструкцією мови.

Вона наказує обчислити вираз, що стоїть праворуч від знака присвоєння, і присвоїти результат змінної, ім'я якої стоїть зліва від знака присвоєння.

Змінна і вираз повинні мати сумісні типи, наприклад дійсний і цілочисловий (але не навпаки). Допускається присвоєння значень даних будь-якого типу, окрім файлового.

Формат інструкції присвоєння:

<Ім’я змінної := <Вираз>;

Замість імені змінної можна вказувати елемент масиву або поле запису. Зазначимо, що знак присвоєння := відрізняється від знака рівності = і має інший зміст.

Знак присвоєння означає, що значення виразу спочатку обчислюється, а потім присвоюється вказаній змінній. Тому за умови, що x є числовою змінною і має певне значення, допустимою і правильною буде така конструкція:

```
x := x + 1;
```

Приклади інструкцій присвоєння для змінних з такими описами:

```
var x, y: real; n: integer; stroka: string;
...
n := 17 * n — 1;  stroka := 'Дата ' + DateToStr(Date);
x := -12.3 * sin(pi / 4); y := 23.789E+3;
```

5.3. Інструкція переходу

Інструкція переходу призначена для змінення звичайного порядку виконання інструкцій програми. Вона використовується у випадках, коли після виконання якоїсь інструкції необхідно виконати не наступну по порядку, а яку-небудь іншу інструкцію. При цьому для здійснення переходу

інструкція, якій передається керування, має бути позначена міткою. Мітка, що стоїть перед інструкцією, відокремлюється від неї двокрапкою.

Нагадаємо, що міткою може бути ідентифікатор або ціле число без знака в діапазоні 0...9999, а всі мітки мають бути заздалегідь оголошені у розділі оголошення міток того блока процедури, функції або програми, в якому ці мітки використуються.

Формат інструкції переходу:

goto <Мітка>;

Приклад використовування інструкції переходу:

Label m1; … goto m1;

...

m1: <Инструкція>;

Передавати керування за допомогою інструкції переходу можна інструкціям, які в тексті програми вище або нижче інструкції переходу.

Забороняється передавати керування інструкціям, що знаходяться усередині структурованих инструкцій, а також інструкціям, що знаходяться в інших блоках (процедурах, функціях).

5.4. Порожня інструкція

Порожня інструкція є крапкою з комою і може бути розташована у будь-якому місці програми, де допускається наявність інструкції. Як і інші инструкції, порожня інструкція може бути позначена міткою. Порожня інструкція не виконує ніяких дій і може бути використана для передачі керування в кінець циклу або складової інструкції.

5.5. Інструкція виклику процедури

Інструкція виклику процедури призначена для активізації стандартної описаної користувачем процедури і є ім'ям цієї процедури і списком переданих їй параметрів.

5.6. Структуровані інструкції

Структуровані інструкції є конструкціями, побудованими за певними правилами з інших інструкцій.

До структурованих інструкцій належать:

— складова інструкція;
— умовна інструкція;
— інструкція вибору;
— інструкції циклу;
— інструкція доступу.

Складена інструкція

Складена інструкція є групою з довільної кількості будь-яких інструкцій, відокремлених одна від одної крапкою з комою і взятих в операторні дужки begin і end. Формат складеної інструкції:

begin <Оператор1>; ...; <ОператорN>; end;

Незалежно від кількості вхідних до неї інструкцій склена інструкція є єдиним цілим і може розташовуватися у будь-якому місці програми, де допускається наявність інструкції.

Найбільш часто складена інструкція використовується в умовних інструкціях та інструкціях циклу. Приклад складеної інструкції:

```
begin
Beep; Edit1.Text := 'Строка'; Exit;
```

Наведена складена інструкція може бути використана в умовній інструкції при перевірці здійснимості деякої умови, скажімо, для показу дій при виникненні помилки.

Складені інструкції можуть вкладатися одна в одну, при цьому глибина вкладення складених інструкцій не обмежена.

5.7. Умовна інструкція

Умовна інструкція забезпечує виконання або невиконання деяких инструкцій залежно від дотримання певних умов. Умовна інструкція в загальному випадку призначена для організації розгалуження на два напрямки і такий формат:

if <Умова> then <Інструкція1> [else <Інструкція2>];

Умова є виразом логічного типу. Інструкція працює таким чином: якщо Умова істинна (має значення True), то виконується Інструкція 1, інакше — Інструкція 2.

Обидві інструкції можуть бути складовими.

Допускається запис умовної інструкції у скороченій формі, коли слова else і Інструкція2 відсутні. У цьому випадку при невиконанні умови керування відразу передається інструкції, яка іде за умовною.

Для організації розгалуження на три і більш напрямків можна використовувати декілька умовних інструкцій, вкладених одна в одну.

При цьому кожне слово else відповідає тому then, яке безпосередньо йому передує.

Через можливу плутанину слід уникати великої глибини вкладення умовних інструкцій.

Приклади умовних інструкцій:

```
if x > 0 then x := x + 1 else x := 0;
if q = 0 then a := 1;
```

5.8. Інструкція вибору

Інструкція вибору є узагальненням умовної інструкції і дозволяє зробити вибір з довільної кількості наявних варіантів, тобто організувати розгалуження на довільну кількість напрямків. Ця інструкція складається з виразу, який називається селектором, списку варіантів і необов'язкової гілки else, що має той же зміст, що й в умовній інструкції.

Формат оператора вибору:

case <Вираз-селектор> <Список > : <Інструкція1>;
...
<Список N> : <Інструкція N> else <Інструкція>;
end;

Вираз-селектор має бути порядкового типу. Кожний з варіантів вибору (від Списку1 до СпискуN) є списком констант, відокремлених двокрапкою від даного варіанта інструкції. Він може бути складовим. Список констант вибору складається з довільної кількості унікальних значень і діапазонів, відокремлених один від одного комами. Межі діапазону записуються двома константами через роздільник "..". Тип констант має збігатися з типом виразу селектора.

Інструкція вибору виконується таким чином:

1. Обчислюється значення виразу-селектора.

2. Проводиться послідовний перегляд варіантів на предмет збігу значенння виразу селектора з константами і значеннями з діапазонів відповідного типу.

3. Якщо для чергового варіанта цей пошук успішний, то виконується інструкція цього варіанта, після чого виконання інструкції вибору закінчується.

4. Якщо всі перевірки виявилися безуспішними, то виконується інструкція, що стоїть після слова else (за його наявності).

Приклад інструкції вибору:

```
case DayNumber
1 .. 5 : strDay:='Робочий день';
6, 7 : strDay:='Вихідний день'
else strDay:=''; end;
```

Залежно від значення цілочислової змінної DayNumber, що містить номер дня тижня, присвоюється відповідне значення рядкової змінної strDay.

5.9. Інструкції циклу

Інструкції циклу призначено для організації циклів (повторів).

Цикл являє собою послідовність інструкцій, яка може виконуватися більше одного разу.

Групу повторюваних інструкцій називають тілом циклу.

Для побудови циклу можна застосовувати умовну інструкцію і інструкцію переходу, розглянуті раніше.

Проте у більшості випадків зручно використовувати інструкції циклу.

Всього є три види інструкцій циклу:

— з параметром;

— з постумовою;

— з передумовою.

Зазвичай, якщо кількість повторів відома наперед, то застосовується інструкція з параметром, інакше — інструкції з пост– або передумовою.

Виконання інструкції циклу будь-якого вигляду може бути перервано за допомогою инструкції переходу goto або призначеної для цих цілей процедури без параметрів Break, яка передає керування, що іде за інструкцією циклу.

За допомогою процедури без параметрів continue можна задати дострокове завершення чергового повторення тіла циклу, що рівнозначно передачі керування в кінець тіла циклу.

Інструкції циклів можуть бути вкладеними одна в одну.

Інструкція циклу з параметром має два можливі формати:

— for <Параметр> := <Вираз1> to <Вираз2> do <Інструкція>;

— for <Параметр> := <Вираз1> downto <Вираз2> do <Інструкція>.

Параметр циклу (Параметр) є змінною порядкового типу, яка має бути визначена у тому ж блоці, де знаходиться інструкція циклу.

Вираз1 і Вираз2 є, відповідно, початковим і кінцевим значеннями параметра циклу і повинні мати тип, сумісний з типом параметра циклу.

Інструкція циклу (Інструкція) забезпечує виконання тіла циклу, якою є інструкція після слова do, до повного перебору з відповідним кроком усіх значень параметра циклу від початкового до кінцевого.

Крок параметра завжди дорівнює 1 для першого формату циклу, і –1 — для другого, тобто значення параметра послідовно збільшується (for...to) або зменшується (for...downto) на одиницю при кожному повторенні циклу. Цикл може не виконатися жодного разу, якщо для циклу for...to значення початкового виразу більше кінцевого, а для циклу for...downto, навпаки, значення початкового виразу менше кінцевого.

Розглянемо приклади циклів з параметрами:

```
var n, k: integer;
...
s := 0;
for n := 1 to 10 do s := s + m[n];
for k := 0 to 2 do
for n := 5 to 10 do begin arr1[k, n] := 0;
arr2[k, n] := 1; end;
```

У першому циклі виконується розрахунок суми десяти значень масиву m. В іншому випадку два цикли вкладено один в один, і в них перераховуються значення елементів двовимірних масивів arr1 і arr2.

Інструкцію циклу з постумовою доцільно використовувати у випадках, коли тіло циклу необхідно виконати не менше одного разу і наперед невідома загальна кількість повторень циклу.

Інструкція циклу з постумовою має такий формат:

repeat
<Інструкція1>;
...
<ІнструкціяN>; until <Умова>;

Умова є виразом логічного типу.

Інструкції, укладені між словами repeat і until, складають тіло циклу і виконуються до тих пір, поки логічний вираз Умова не набула значення True, тобто тіло циклу повторюється при значенні Умови, що дорівнює False.

Оскільки Умова перевіряється тільки в кінці циклу інструкції, тіла циклу виконуються мінімум один раз.

У тілі циклу може знаходитися довільна кількість інструкцій без операторних дужок begin і end.

Принаймні одна з інструкцій тіла циклу повинна впливати на значенння Умова, інакше відбудеться зациклення.

Для ілюстрації циклу з постумовою наведемо приклад розрахунку суми десяти значень масиву m:

```
var x: integer; sum: real;
m: array[1..10] real;
...
x := 1; sum := 0;
repeat
sum := sum + m[x]; x := x + 1;
until(x < 10);
```

Інструкцію циклу з передумовою доцільно використовувати у випадках, коли rskmrscnm повторень тіла циклу наперед невідомо і тіло циклу може жодного разу не виконуватися.

Ця інструкція аналогічна інструкції repeat...until з тією лише різницею, що умова перевіряється на початку оператора.

Формат інструкції циклу з предумовою:

while <Умова> do <Інструкція>;

Інструкція тіла циклу виконується до тих пір, поки логічний вираз Умова не набуде значення False, тобто на відміну від циклу з постумовою цей цикл виконується при значенні логічного виразу True.

Як приклад розглянемо розрахунок суми десяти значень масиву m:

```
var x: integer;
sum: real;
m: array[1..10] real;
...
x:=1; sum:=0;
while x <= 10 do begin
sum := sum + m[x];
x := x + 1;
…
end;
```

Якщо перед першим виконанням циклу умова не задовольняється (значення логічного виразу Умова дорівнює False), то тіло циклу не виконується жодного разу, і відбувається перехід на інструкцію, яка іде за інструкцією циклу.

5.11. Інструкція доступу

Інструкція доступу призначена для зручної і швидкої роботи зі складовими частинами об'єктів, у тому числі з полями записів. Нагадаємо, що для звернення до поля запису необхідно указувати ім'я запису й ім'я цього поля, розділені крапкою. Аналогічним шляхом утворюється ім'я складової частини будь-якого об'єкта, наприклад форми або кнопки. Інструкція доступу має такий основний формат:

with <Ім’я об’єкта> do <Інструкція>

В інструкції, розташованій після слова do, для звернення до складової частини об'єкта можна не указувати ім'я цього об'єкта, яке вже задане після слова with. Наведемо приклад двох варіантів звернення до складових частин об'єкта:

```
// Складові імена пишуться повністю
…
Form1.Canvas.Pen.Color := clRed;
Form1.Canvas.Pen.Width := 5;
Form1.Canvas.Rectangle(10, 10, 100, 100);
…
// Використовування інструкції доступу
…
with Form1.Canvas do begin
Pen.Color := clRed;
Pen.Width := 5;
Rectangle(10, 10, 100, 100);
…
end;
```

6. ВИКОРИСТАННЯ ВІЗУАЛЬНИХ КОМПОНЕНТІВ

6.1. Сторінки візуальних компонентів

Для створення інтерфейсу додатків система Delphi пропонує обширний набір візуальных компонентів.

Розглянемо основні, які розташовуються на сторінках Standard, Additional і Win32 Палітри компонентів. Їх називають стандартними, додатковими, 32-розрядними компонентами відповідно. Такий розподіл компонентів виходить швидше з назви сторінок, ніж з їх функціонального призначення або важливості, оскільки грань, наприклад між стандартними і додатковими елементами керування досить нечітка.

Так, кнопки Button і BitBtn, розташовуючись на різних сторінках, практично не відрізняються за функціями.

На сторінці Standard (рис. 6.1) Палітри компонентів знаходяться інтерфейсні компоненти:

- Frames (фрейми);
- MainMenu (головне меню);
- PopupMenu (спливаюче меню);
- Label (напис);
- Edit (однорядковий редактор);
- Memo (багаторядковий редактор);
- Button (стандартна кнопка);
- CheckBox (незалежний перемикач, прапорець);
- RadioButton (залежний перемикач, перемикач);
- ListBox (список);
- ComboBox (поле із списком);
- ScrollBar (смуга прокручування);
- GroupBox (група);
- RadioGroup (група залежних перемикачів);
- Panel (панель);
- ActionList (список дій).

Рис. 6.1. Сторінка Standard Палітри компонентів

Компоненти перераховані послідовно в порядку розташування їх значків на сторінці.

Перший значок відповідає не компоненту, а інструменту (стрілці) відміни вибору компонента на сторінці.

На сторінці Additional (рис. 6.2) Палітри компонентів знаходяться такі компоненти:

— BitBtn (кнопка з рисунком);
— SpeedButton (кнопка швидкого доступу);
— MaskEdit (однорядковий редактор з уведенням за шаблоном);
— StringGrid (таблиця рядків);
— DrawGrid (таблиця);
— Image (графічне зображення);
— Shape (геометрична фігура);
— Bevel (фаска);
— ScrollBox (область прокручування);
— CheckListBox (список прапорців);
— Splitter (подільник);
— StaticText (статичний текст);
— ControlBar (контейнер для панелі інструментів);
— ApplicationEvents (події додатка);
— ValueListEditor (редактор списку значень);
— LabeledEdit (однорядковий редактор з написом);
— ColorBox (комбінований список вибору кольору);
— Chart (діаграма);
— ActionManager (менеджер дій);
— ActionMainMenuBar (головне меню дій);
— ActionToolBar (панель дій);
— XPColorMap (колірна карта XP для меню і панелей інструментів);
— StandardColorMap (стандартна колірна карта для меню і панелей інструментів);
— TwilightColorMap (колірна карта Twilight для меню і панелей інструментів);
— CustomizeDlg (діалог настройки).

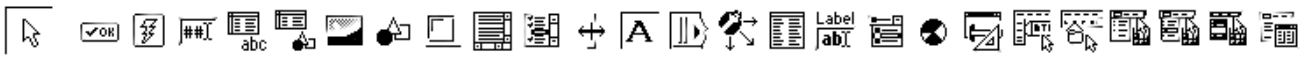

Рис. 6.2. Сторінка Additional Палітри компонентів

На сторінці Win32 (рис. 6.3) Палітри компонентів знаходяться компоненти, які належать до 32-розрядного інтерфейсу Windows:

— TabControl (вкладка);
— PageControl (блокнот);
— ImageList (список графічних зображень);
— RichEdit (повнофункціональний тестовий редактор);

— TrackBar (повзунок);
— ProgressBar (індикатор виконання);
— UpDown (лічильник);
— HotKey (редактор комбінацій "гарячих" клавіш);
— Animate (проглядання відеокліпів);
— DateTimePicker (рядок уведення дати);
— MonthCalendar (календар);
— TreeView (дерево об'єктів);
— ListView (список);
— HeaderControl (подільник);
— StatusBar (рядок стану);
— ToolBar (панель інструментів);
— CoolBar ("крута" панель інструментів);
— PageScroller (прокручування зображень);
— ComboBoxEx (розширений комбінований список);
— XPManifest (декларація XP).

Рис. 6.3. Сторінка компонентів Win32

Розглянемо візуальні компоненти, що найбільш часто використовуються як елементи керування додатків.

6.2. Загальна характеристика візуальних компонентів

У бібліотеці візуальних компонентів VCL для всіх компонентів, у тому числі і для призначених для роботи з даними, базовим є клас TControl. Вони забезпечують основні функціональні атрибути, такі як положення і розміри елемента, його заголовок, колір та інші параметри. Клас TControl включає загальні для візуальних компонентів властивості, події і методи.

У цілому візуальні компоненти можна поділити на дві великі групи: віконні й невіконні елементи керування. Віконний елемент керування є спеціалізованим вікном, призначеним для вирішення конкретної задачі. До таких елементів належать, наприклад, кнопки, текстові поля, смуги прокручування. Для віконних елементів керування базовим класом є TWinControl — прямий нащадок класу TControl.

На отримання фокусу введення віконні елементи керування можуть реагувати двома способами:

— за допомогою курсора редагування;
— за допомогою прямокутника фокусу.

Такі компоненти, як редактори Edit, DBEdit, Memo або DBMemo, при отриманні фокусу введення відображають у своїй області курсор редагування (текстовий курсор).

За замовчуванням курсор редагування має вид миготливої вертикальної лінії і показує поточну позицію вставки з клавіатури символів, що вводяться. Курсор редагування переміщується за допомогою клавіш керування курсором.

Компоненти, не пов'язані з редагуванням тексту, отриманням фокусу введення зазвичай відображають за допомогою пунктирного чорного прямокутника. При цьому, наприклад для кнопки Button, цей прямокутник з'являється навкруги її заголовка, а для списків ListBox і DBListBox прямокутник виділяє вибрану у певний момент часу рядок.

Вибраний рядок може забарвлюватися у який-небудь колір, частіше за все синій. Віконні елементи керування містять дескриптор (визначник) вікна (windowhandle). Дескриптором вікна в операційній системі Windows називається 32-бітова величина, яка однозначно визначає це вікно. Додаток використовує цей визначник для звернення до вікна.

Крім того, віконні елементи керування можуть містити інші елементи керування, які відіграють роль класу-контейнера, що є батьківським стосовно об'єктів, що знаходяться в ньому.

Для невіконних елементів керування базовим є клас TGraphicControl, який походить безпосередньо від класу TControl.

Невіконні елементи керування не можуть одержувати фокус уведення і бути батьками інших елементів, призначених для користувача інтерфейсу. Перевагою невіконних елементів керування порівняно з віконними є менше витрачання ресурсів, оскільки для них не потрібен дескриптор вікна. Невіконними елементами керування є, наприклад, компоненти Label і DBText.

Розглянемо докладніше загальні властивості, події і методи візуальних компонентів, а також клас TStrings, який є базовим класом для операцій з рядковими даними.

6.3. Властивості

Властивості дозволяють керувати зовнішнім виглядом і поведінкою компонентів при проектуванні і виконанні додатка.

Властивості компонентів, доступні при проектуванні додатка, також є доступними при його виконанні.

Разом з тим є властивості, які доступні тільки під час виконання додатка.

Зазвичай більшість значень властивостей компонентів визначаються на етапі проектування за допомогою Інспектора об'єктів, проте для

наочності в наведених нами прикладах властивостям часто присвоюються значення за допомогою інструкції присвоєння.

Властивості можна поділитии на такі групи (у дужках наведено назви груп у вікні Інспектора об'єктів):

— дія (Action);
— операції з базами даних (Database);
— переміщення і поєднання компонентів (Drag, Drop and Docking);
— вхідні (Input);
— контекстна допомога (Help and Hints);
— макет (Layout);
— спадкоємство (Legacy);
— зв'язки (Linkage);
— місцеві (Locale);
— локалізація (Localizable);
— різне (Miscellaneous);
— візуальні (Visual).

У вікні Інспектора об'єктів властивість може відображатися відразу в декількох групах.

Наприклад, властивість Visible одночасно належить до груп Action і Visual, а Caption — до груп Action, Localizable і Visual.

При зміненні значення властивості в одній групі також змінюються значення, що відображаються в інших групах.

Розглянемо загальні властивості візуальних компонентів, описавши властивості (кроме Name) в алфавітному порядку.

Зазначимо, що окремі компоненти мають не всі властивості, які були наведені для інших. Наприклад, редактор Edit не має властивості Caption, а напис Label — властивості ReadOnly.

Властивість Name типу TComponentName указує ім'я компонента, яке програміст використовує для керування компонентом під час виконання додатка.

Зазначимо, що тип TComponentName еквівалентний типу String.

Кожний новий компонент, який розміщують у форму, одержує ім'я за замовчуванням, що автоматично утворюється шляхом додавання до назви компонента його номера в порядку розміщення у форму.

Наприклад, перший однорядковий редактор Edit одержує ім'я Edit1, другий — Edit2.

На етапі розроблення додатка програміст може змінити ім'я компонента за замовчуванням на більш осмислене і таке, що відповідає призначенню компонента.

Є декілька точок зору з приводу присвоєння імен компонентам. Згідно з однією з них ім'я рекомендується складати згідно з призначенням компонента і його назви.

Іншим варіантом буде зазначення в імені замість назви компонента його префікса.

Префікс є скороченням назви, наприклад, для однорядкового редактора Edit префікс може бути edt, для напису Label — lbl, для форми Form — fm.

Таким чином, однорядковий редактор, призначений для введення прізвища співробітника, можна назвати NameEdit або edtName.

Обидва способи є допустимими, і на практиці кожний розробник використовує той, який для нього є найбільш зручним, або взагалі називає компоненти так, як йому хочеться.

Для наочності використовуватимемо в наших прикладах як імена візуальных і невізуальних компонентів їх імена за замовчуванням, наприклад Label1, Edit2 або Button3.

При динамічному створенні компонентів під час виконання додатка вони теж автоматично одержують імена за замовчуванням.

Розробник може змінити ім'я нового компонента, програмно встановивши потрібне значення його властивості Name.

Наведемо приклад створення компонента під час виконання програми:

```
with Edit.Create(Self) do begin
Parent := Form1; Name := 'edtName';
Text := 'Іванов О. Ю.'; Left := 100; Top := 60; end;
```

Тут динамічно створюється однорядковий редактор Edit, який одержує ім'я edtName.

Новому редактору визначаються текстове значення і координати його розміщення в контейнері — власнику цього компонента.

Власником нового редактора є форма Form1.

Організувати сумісне використовування декількох взаємозв'язаних елементів керування можна за допомогою спеціального компонента ActionList.

Він призначений для централізованого керування різними елементами, наприклад такими, як кнопка Button і пункт меню MenuItem.

Зв'язок між елементом керування і об'єктом дії, що міститься в компоненті ActionList, здійснюється через властивість Action типу TBasicAction елемента керування. Розглянемо на прикладі, як встановлюється зв'язок з об'єктом дії:

```
procedure TForm1.FormCreate(Sender: TObject);
begin Button1.Action := Action1;
end;
```

Тут кнопка Button1 пов'язана з об'єктом дії Action1. При натисненні кнопки Button1 буде викликаний не обробник події OnClick кнопки, а процедура оброблення події OnExecute об'єкта Action1.

Властивість Align типу TAlign визначає спосіб вирівнювання компонента усередині контейнера, в якому він знаходиться. Найчастіше таким контейнером є форма (Form) або панель (Panel).

Вирівнювання використовується у випадках, коли необхідно, щоб будь-який інтерфейсний елемент займав певне положення відносно контейнера, до якого він входить незалежно від змінення розмірів останнього.

Властивість Align може набувати таких значень:

— alNone — вирівнювання не використовується, компонент за замовченням знаходиться на тому місці, куди був поміщений при розробленні додатка;

— alTop — компонент переміщується у верхню частину контейнера, висота компонента не змінюється, ширина при цьому дорівнює ширині контейнера;

— alBottom — компонент переміщується в нижню частину контейнера, висота компонента не змінюється, а ширина при цьому дорівнює ширині контейнера (аналогічно дії alTop);

— alLeft — компонент переміщується в ліву частину контейнера, ширина компонента не змінюється, а його висота при цьому дорівнює висоті контейнера;

— alRight — компонент переміщується у праву частину контейнера, ширина компонента не змінюється, а його висота при цьому дорівнює висоті контейнера (аналогічно дії alLeft);

— alClient — компонент займає всю поверхню контейнера;

— alCustom — розміри і положення компонента в контейнері встановлюються розробником.

Наприклад, вирівнювання панелі відносно форми виконується за допомогою такого коду:

```
Panel1.Align := alClient;
```

Властивість Caption типу TCaption містить рядок для заголовка компонента. Зазначимо, що тип TCaption дорівнює типу String.

Окремі символи в заголовку можуть бути підкреслені, вони позначають комбінації клавіш швидкого доступу: натиснення клавіші із указаним символом при натиснутій клавіші <Alt> викликає ту ж дію, що і клацання мишею на елементі керування з цим заголовком. Для визначення комбінації клавіш необхідно поставити в заголовку перед відповідним символом знак &.

Наприклад:

```
CheckBox1.Caption := 'мммммммм'; <Alt>+<3>
RadioGroup1.Caption := '&Conditions'; <Alt>+<C>
```

При реагуванні на комбінації клавіш Windows ураховує розташування клавіатури, при цьому користувач повинен не забувати перемикати мову,

наприклад з української на анлійську, і навпаки. Властивість Color типу TColor визначає колір фону (поверхні) компонента. Тип TColor описаний таким чином:

type TColor = -(COLOR ENDCOLORS + 1) .. $02FFFFFF;

Значення властивості Color є чотирибайтовим шістнадцятковим числом. Старший байт указує палітру і зазвичай має значення $00, що відповідає відображенню кольору, найближчого до того, що задається властивістю Color.

Молодші три байти задають RGB-інтенсивності (інтенсивності базових червоного, зеленого і синього кольорів), які при змішуванні дають колір, що потребується. Коли значення байта, що містить код інтенсивності, дорівнює $FF, відповідний базовий колір має максимальну інтенсивність, якщо значення байта дорівнює $00, то відповідний базовий колір вимкнено. Відсутність базових кольорів приводить до чорного кольору, а їх максимальна інтенсивність утворює білий колір.

Таким чином, чорному кольору відповідає код $000000, білому — $FFFFFF, червоному — $0000FF, зеленому — $00FF00, синьому — $FF0000. Часто зручно задавати кольори за допомогою констант. Колір, що відображається, залежить від параметрів відеокарти і монітора, у першу чергу від встановленого колірного розрізнення.

При використовуванні констант відображається колір, найближчий до вказаного константою. Усі константи, окрім clDkGray і clLtGray, можна вибирати за допомогою Інспектора об'єктів. Додатково під час виконання додатка можна використовувати константи clDkGray і clLtGray, які дублюють значення clGray і clSilver відповідно. Властивість Visible типу Boolean керує видимістю компонента. Якщо для нього встановлено значення True, то компонент видно користувачу, при значенні False компонент прихований від користувача. Зазначимо, що навіть якщо компонента не видно, ним можна керувати програмно. Наприклад:

Edit1.Visible := True; Edit2.Visible := False;

6.4. Події

Візуальні компоненти здатні генерувати і обробляти достатньо велику кількість (декілька десятків) подій різних видів.

До найбільш загальних груп подій можна віднести такі:

— вибір елемента керування;
— переміщення покажчика миші;
— обертання колеса миші;
— натиснення клавіш;
— отримання і втрата елементом керуванням фокусом введення;
— переміщення об'єктів методом drag-and-drop (перетягуванням).

7. ПРИКЛАД ОСНОВНИХ ЕКРАННИХ ФОРМ З ВИКОРИСТАННЯМ ВІЗУАЛЬНИХ КОМПОНЕНТІВ У ПРОГРАМНОМУ КОМПЛЕКСІ ДЛЯ РОЗРАХУНКУ РЕЖИМІВ РІЗАННЯ

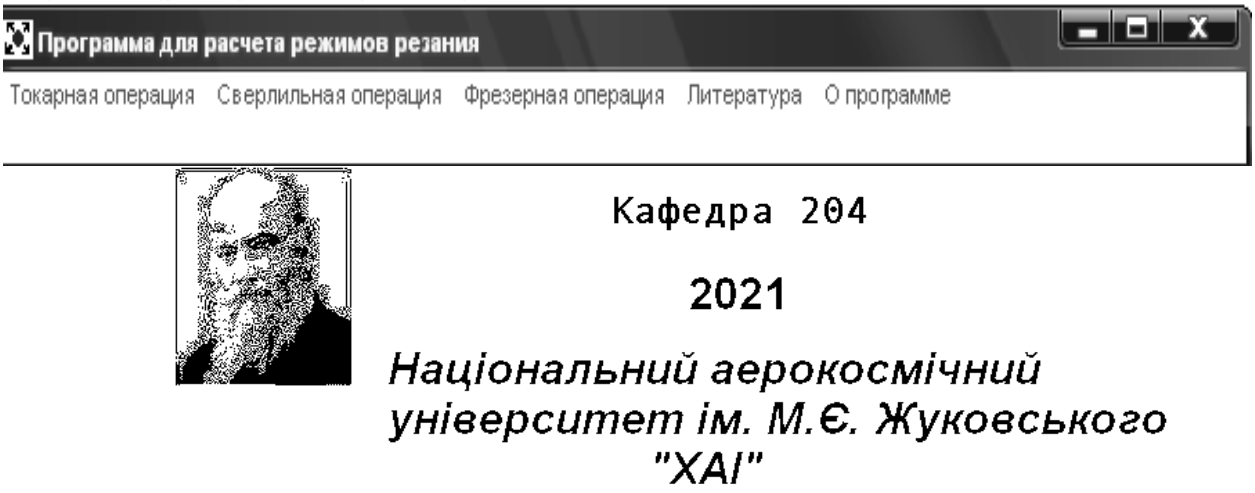

Рис. 7.1. Головна екранна форма меню програми

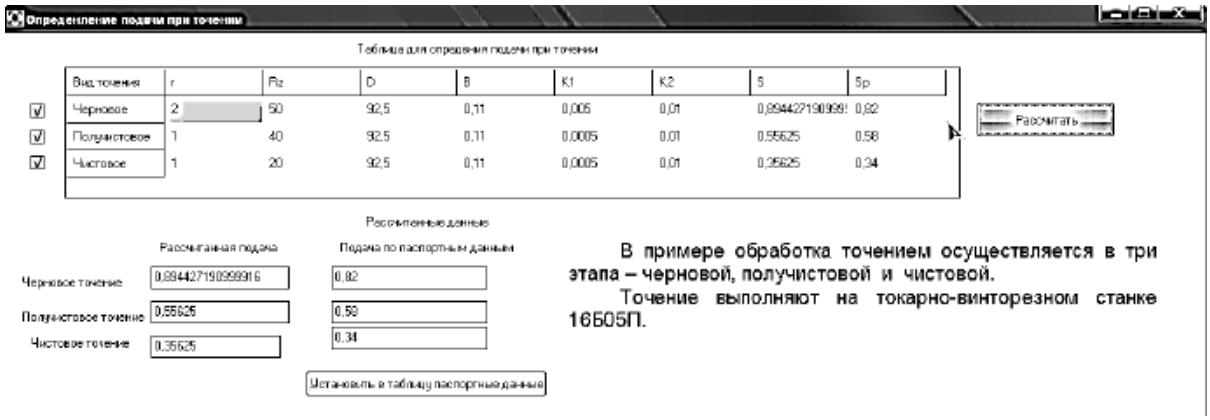

Рис. 7.2. Екранна форма для розрахунків подач при точінні

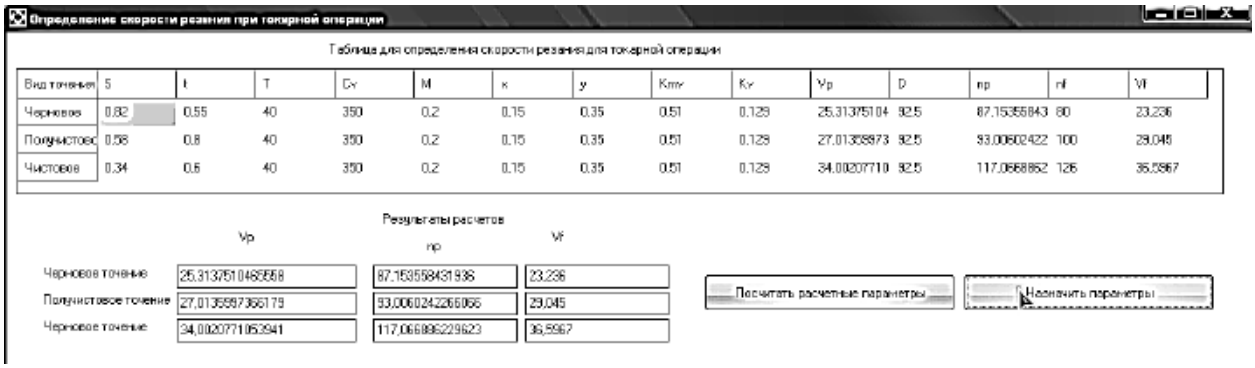

Рис. 7.3. Екранна форма для розрахунків швидкості при точінні

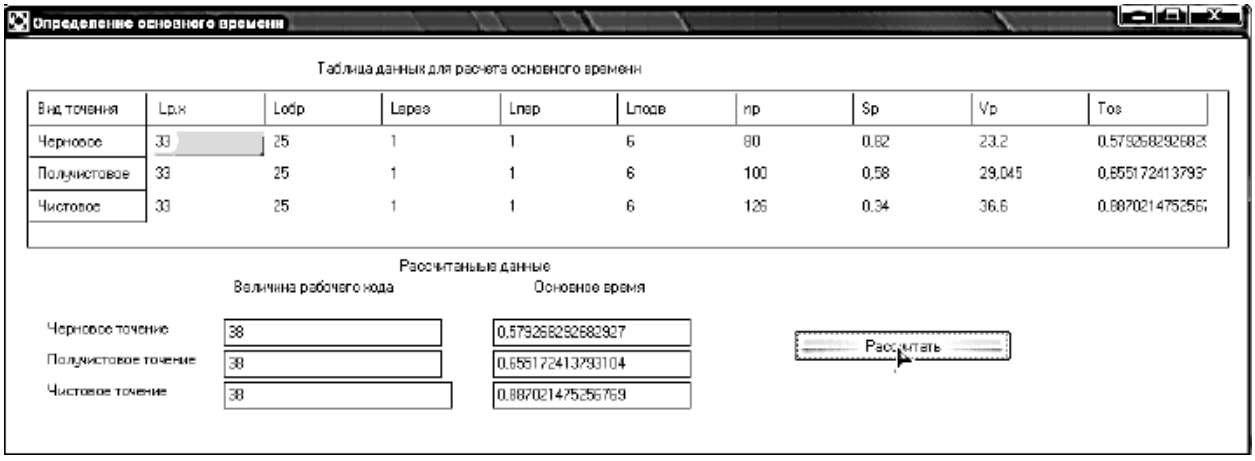

Рис. 7.4. Екранна форма для розрахунків основного часу при точінні

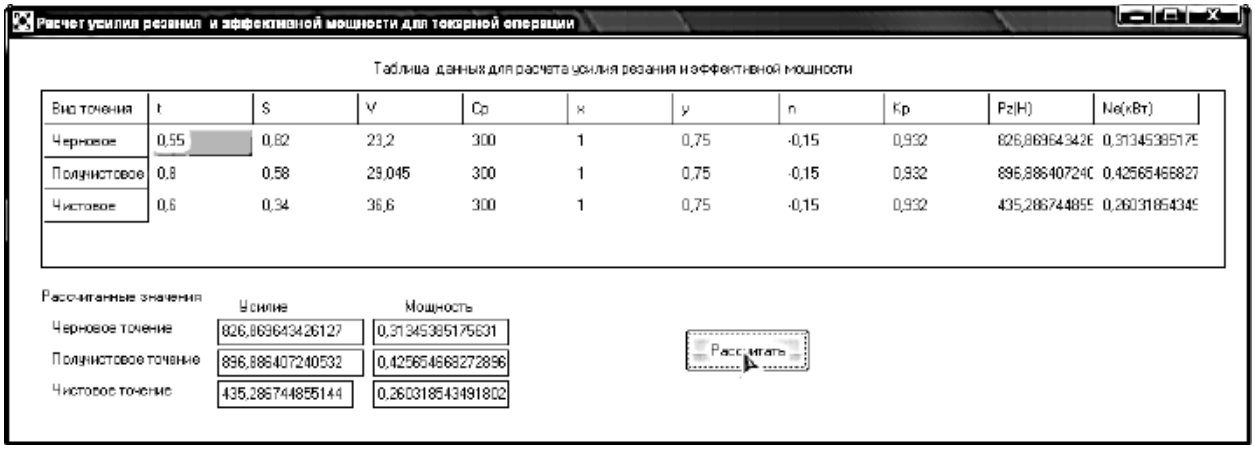

Рис. 7.5. Екранна форма для розрахунків ефективної потужності при точінні

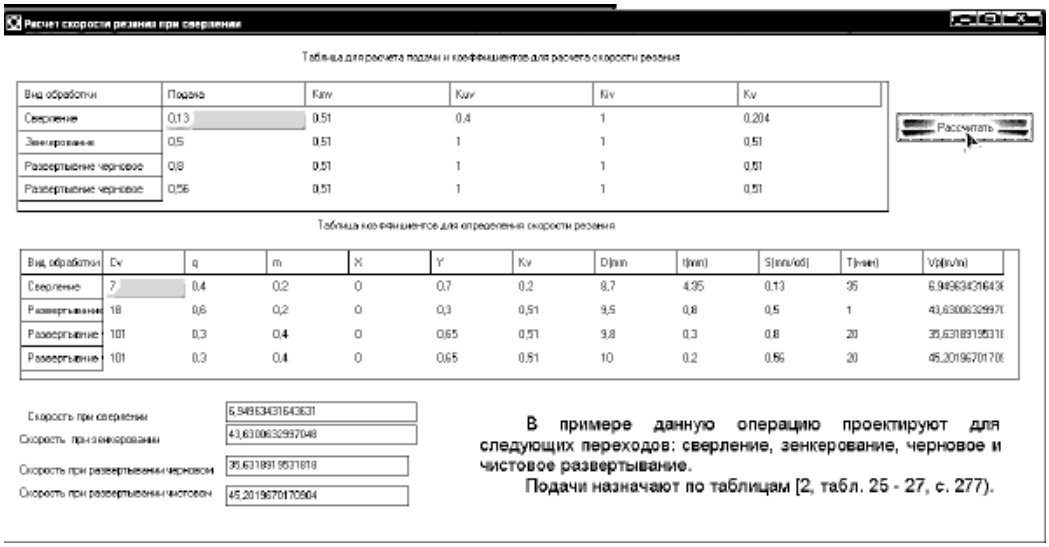

Рис. 7.6. Екранна форма для розрахунків швидкості при свердлінні

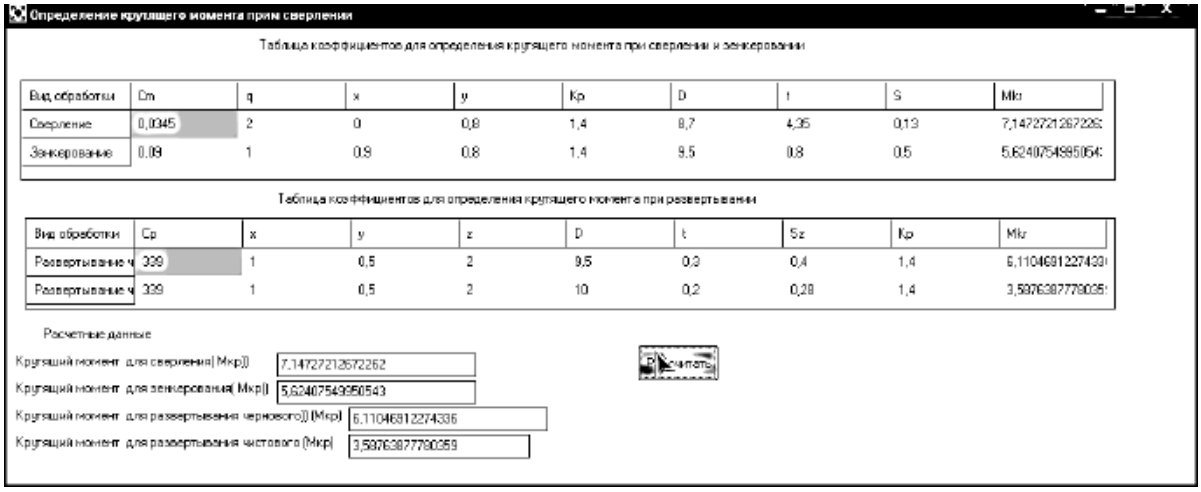

Рис. 7.7. Екранна форма для розрахунків крутного моменту при свердлінні

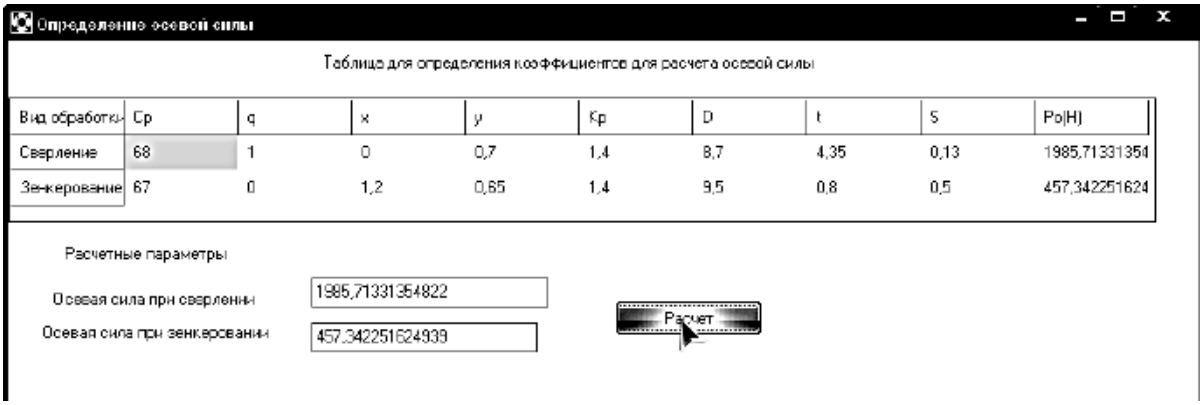

Рис. 7.8. Екранна форма для розрахунків осьової сили при свердлінні

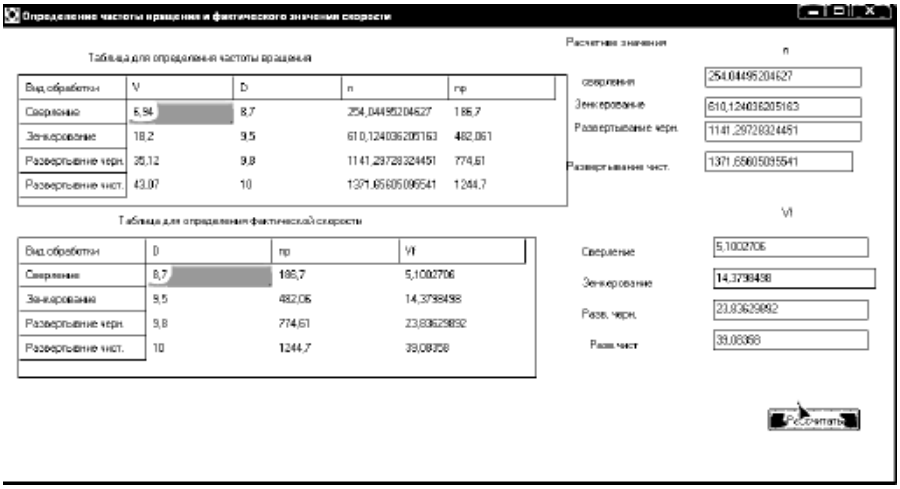

Рис. 7.9. Екранна форма для розрахунків фактичного значення швидкості при свердлінні

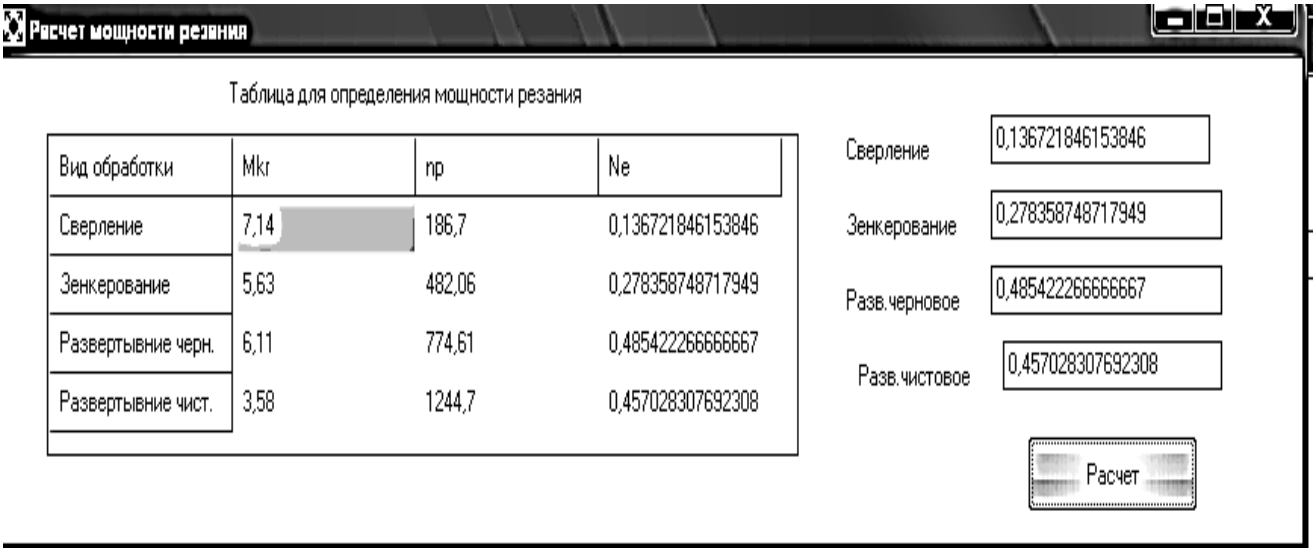

Рис. 7.10. Екранна форма для розрахунків фактичного значення потужності при свердлінні

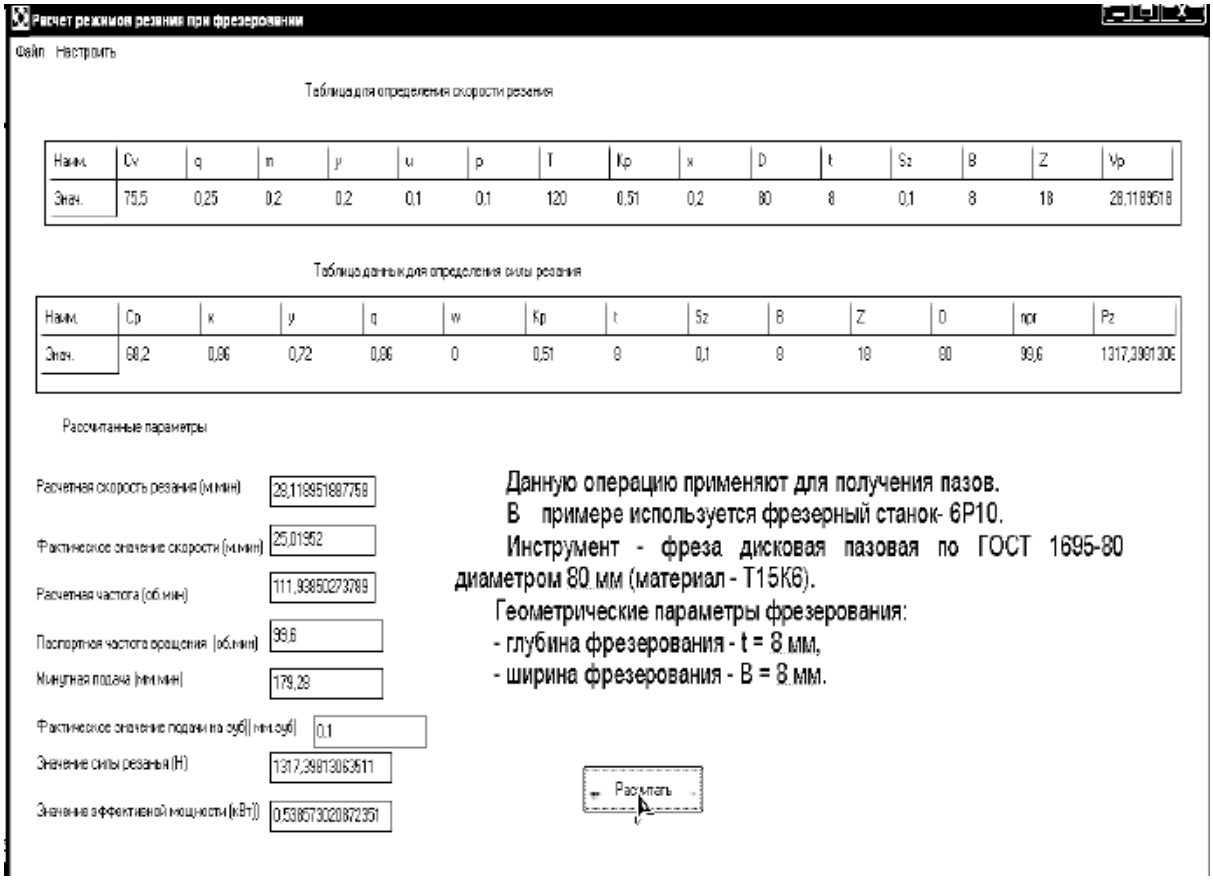

Рис. 7.11. Екранна форма для розрахунків режимів різання при фрезеруванні

8. ПЕРВИННИЙ ТЕКСТ ПРОГРАМИ ОДНОГО З МОДУЛІВ ДЛЯ РОЗРАХУНКІВ РЕЖИМІВ РІЗАННЯ

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, Buttons, ExtCtrls;

type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    Label1: TLabel;
    StringGrid2: TStringGrid;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Label4: TLabel;
    Edit2: TEdit;
    BitBtn1: TBitBtn;
    Label5: TLabel;
    Edit3: TEdit;
    Label6: TLabel;
    Edit4: TEdit;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);

  private
    { Private declarations }
  public

    { Public declarations }
  end;

var
  Form1: TForm1;
  pod:array [1..4] of real;
```

```
  Kmv:array [1..4] of real;
  Kuv:array [1..4] of real;
  Kiv:array [1..4] of real;
  Kv:array [1..4] of real;
  Cv:array [1..4] of real;
     q:array [1..4] of real;
      m:array [1..4] of real;
        X:array [1..4] of real;
          y:array [1..4] of real;
            Kv_:array [1..4] of real;
             D:array [1..4] of real;
                t:array [1..4] of real;
                  S:array [1..4] of real;
                     T_:array [1..4] of real;
                      Vp:array [1..4] of real;

      Vp1:real;
      Vp2:real;
      Vp3:real;
      Vp4:real;
      i:integer;

 implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
   pod[1]:=0.13;     pod[2]:=0.5;    pod[3]:=0.8;    pod[4]:=0.56;
   Kmv[1]:=0.51;     Kmv[2]:=0.51;   Kmv[3]:=0.51;   Kmv[4]:=0.51;
   Kuv[1]:=0.4;      Kuv[2]:=1;      Kuv[3]:=1;   Kuv[4]:=1;
   Kiv[1]:=1;        Kiv[2]:=1;      Kiv[3]:=1;   Kiv[4]:=1;
   Kv[1]:=0.204;     Kv[2]:=0.51;    Kv[3]:=0.51;   Kv[4]:=0.51;

   Cv[1]:=7;         Cv[2]:=18;       Cv[3]:=101;      Cv[4]:=101;
   q[1]:=0.4;        q[2]:=0.6;       q[3]:=0.3;       q[4]:=0.3;
   m[1]:=0.2;        m[2]:=0.2;       m[3]:=0.4;       m[4]:=0.4;
   X[1]:=0;          X[2]:=0;         X[3]:=0 ;        X[4]:=0;
   y[1]:=0.7;        y[2]:=0.3;       y[3]:=0.65;      y[4]:=0.65;
   Kv_[1]:=0.2;      Kv_[2]:=0.51;    Kv_[3]:=0.51;  Kv_[4]:=0.51;
   D[1]:=8.7;        D[2]:=9.5;       D[3]:=9.8;       D[4]:=10;
   t[1]:=4.35;       t[2]:=0.8;       t[3]:=0.3;       t[4]:=0.2;
```

```
 S[1]:=0.13;       S[2]:=0.5;        S[3]:=0.8;       S[4]:=0.56;
 T_[1]:=35;        T_[2]:=1;         T_[3]:=20;          T_[4]:=20;
 Vp[1]:=6.94;      Vp[2]:=18.2;      Vp[3]:=35.1;     Vp[4]:=43.1;
            …
StringGrid1.leftcol:=6;
StringGrid1.toprow:=5;
StringGrid1.colcount:=6;
StringGrid1.rowcount:=5;
StringGrid1.cells[0,0]:='';
StringGrid1.cells[0,1]:='';
StringGrid1.cells[0,2]:='';
StringGrid1.cells[0,3]:='';
StringGrid1.cells[0,4]:='';
 StringGrid1.cells[1,0]:=';
StringGrid1.cells[2,0]:='Kmv';
StringGrid1.cells[3,0]:='Kuv';
StringGrid1.cells[4,0]:='Kiv';
StringGrid1.cells[5,0]:='Kv';
StringGrid1.cells[1,1]:=floattostr(pod[1]);
StringGrid1.cells[1,2]:=floattostr(pod[2]);
StringGrid1.cells[1,3]:=floattostr(pod[3]);
StringGrid1.cells[1,4]:=floattostr(pod[4]);
StringGrid1.cells[2,1]:=floattostr(Kmv[1]);
StringGrid1.cells[2,2]:=floattostr(Kmv[2]);
StringGrid1.cells[2,3]:=floattostr(Kmv[3]);
StringGrid1.cells[2,4]:=floattostr(Kmv[4]);
StringGrid1.cells[3,1]:=floattostr(Kuv[1]);
StringGrid1.cells[3,2]:=floattostr(Kuv[2]);
StringGrid1.cells[3,3]:=floattostr(Kuv[3]);
StringGrid1.cells[3,4]:=floattostr(Kuv[4]);
StringGrid1.cells[4,1]:=floattostr(Kiv[1]);
StringGrid1.cells[4,2]:=floattostr(Kiv[2]);
StringGrid1.cells[4,3]:=floattostr(Kiv[3]);
StringGrid1.cells[4,4]:=floattostr(Kiv[4]);
StringGrid1.cells[5,1]:=floattostr(Kv[1]);
StringGrid1.cells[5,2]:=floattostr(Kv[2]);
StringGrid1.cells[5,3]:=floattostr(Kv[3]);
StringGrid1.cells[5,4]:=floattostr(Kv[4]);
                        …
StringGrid2.leftcol:=12;
StringGrid2.toprow:=5;
StringGrid2.colcount:=12;
```

```
StringGrid2.rowcount:=5;
                …
StringGrid2.cells[0,2]:='';
StringGrid2.cells[0,3]:='';
StringGrid2.cells[0,4]:='';
StringGrid2.cells[1,0]:='Cv';
StringGrid2.cells[2,0]:='q';
StringGrid2.cells[3,0]:='m';
StringGrid2.cells[4,0]:='X';
StringGrid2.cells[5,0]:='Y';
StringGrid2.cells[6,0]:='Kv';
StringGrid2.cells[7,0]:='D(mm';
StringGrid2.cells[8,0]:='t(mm)';
                      …
StringGrid2.cells[11,0]:='Vp(m/m)';
                      …
StringGrid2.cells[1,1]:=floattostr(Cv[1]);
StringGrid2.cells[1,2]:=floattostr(Cv[2]);
StringGrid2.cells[1,3]:=floattostr(Cv[3]);
StringGrid2.cells[1,4]:=floattostr(Cv[4]);
StringGrid2.cells[2,1]:=floattostr(q[1]);
StringGrid2.cells[2,2]:=floattostr(q[2]);
StringGrid2.cells[2,3]:=floattostr(q[3]);
StringGrid2.cells[2,4]:=floattostr(q[4]);
StringGrid2.cells[3,1]:=floattostr(m[1]);
StringGrid2.cells[3,2]:=floattostr(m[2]);
StringGrid2.cells[3,3]:=floattostr(m[3]);
StringGrid2.cells[3,4]:=floattostr(m[4]);
StringGrid2.cells[4,1]:=floattostr(X[1]);
StringGrid2.cells[4,2]:=floattostr(X[2]);
StringGrid2.cells[4,3]:=floattostr(X[3]);
StringGrid2.cells[4,4]:=floattostr(X[4]);
StringGrid2.cells[5,1]:=floattostr(y[1]);
StringGrid2.cells[5,2]:=floattostr(y[2]);
StringGrid2.cells[5,3]:=floattostr(y[3]);
StringGrid2.cells[5,4]:=floattostr(y[4]);
StringGrid2.cells[6,1]:=floattostr(Kv_[1]);
StringGrid2.cells[6,2]:=floattostr(Kv_[2]);
StringGrid2.cells[6,3]:=floattostr(Kv_[3]);
StringGrid2.cells[6,4]:=floattostr(Kv_[4]);
StringGrid2.cells[7,1]:=floattostr(D[1]);
StringGrid2.cells[7,2]:=floattostr(D[2]);
```

```
  StringGrid2.cells[7,3]:=floattostr(D[3]);
  StringGrid2.cells[7,4]:=floattostr(D[4]);
  StringGrid2.cells[8,1]:=floattostr(t[1]);
  StringGrid2.cells[8,2]:=floattostr(t[2]);
  StringGrid2.cells[8,3]:=floattostr(t[3]);
  StringGrid2.cells[8,4]:=floattostr(t[4]);
  StringGrid2.cells[9,1]:=floattostr(S[1]);
  StringGrid2.cells[9,2]:=floattostr(S[2]);
  StringGrid2.cells[9,3]:=floattostr(S[3]);
  StringGrid2.cells[9,4]:=floattostr(S[4]);
  StringGrid2.cells[10,1]:=floattostr(T_[1]);
  StringGrid2.cells[10,2]:=floattostr(T_[2]);
  StringGrid2.cells[10,3]:=floattostr(T_[3]);
  StringGrid2.cells[10,4]:=floattostr(T_[4]);
  StringGrid2.cells[11,1]:=floattostr(Vp[1]);
  StringGrid2.cells[11,2]:=floattostr(Vp[2]);
  StringGrid2.cells[11,3]:=floattostr(Vp[3]);
  StringGrid2.cells[11,4]:=floattostr(Vp[4]);

   end;

procedure TForm1.BitBtn1Click(Sender: TObject);
var i:integer;
Vp1:real;
begin
pod[1]:=strtofloat(StringGrid1.cells[1,1]);
pod[2]:=strtofloat(StringGrid1.cells[1,2]);
pod[3]:=strtofloat(StringGrid1.cells[1,3]);
pod[4]:=strtofloat(StringGrid1.cells[1,4]);
Kmv[1]:=strtofloat(StringGrid1.cells[2,1]);
Kmv[2]:=strtofloat(StringGrid1.cells[2,2]);
Kmv[3]:=strtofloat(StringGrid1.cells[2,3]);
Kmv[4]:=strtofloat(StringGrid1.cells[2,4]);
Kuv[1]:=strtofloat(StringGrid1.cells[3,1]);
Kuv[2]:=strtofloat(StringGrid1.cells[3,2]);
Kuv[3]:=strtofloat(StringGrid1.cells[3,3]);
Kuv[4]:=strtofloat(StringGrid1.cells[3,4]);
Kiv[1]:=strtofloat(StringGrid1.cells[4,1]);
Kiv[2]:=strtofloat(StringGrid1.cells[4,2]);
Kiv[3]:=strtofloat(StringGrid1.cells[4,3]);
Kiv[4]:=strtofloat(StringGrid1.cells[4,4]);
Kv[1]:=strtofloat(StringGrid1.cells[5,1]);
```

```
Kv[2]:=strtofloat(StringGrid1.cells[5,2]);
Kv[3]:=strtofloat(StringGrid1.cells[5,3]);
Kv[4]:=strtofloat(StringGrid1.cells[5,4]);
                    …
Cv[1]:=strtofloat(StringGrid2.cells[1,1]);
Cv[2]:=strtofloat(StringGrid2.cells[1,2]);
Cv[3]:=strtofloat(StringGrid2.cells[1,3]);
Cv[4]:=strtofloat(StringGrid2.cells[1,4]);
q[1]:=strtofloat(StringGrid2.cells[2,1]);
q[2]:=strtofloat(StringGrid2.cells[2,2]);
q[3]:=strtofloat(StringGrid2.cells[2,3]);
q[4]:=strtofloat(StringGrid2.cells[2,4]);
m[1]:=strtofloat(StringGrid2.cells[3,1]);
m[2]:=strtofloat(StringGrid2.cells[3,2]);
m[3]:=strtofloat(StringGrid2.cells[3,3]);
m[4]:=strtofloat(StringGrid2.cells[3,4]);
X[1]:=strtofloat(StringGrid2.cells[4,1]);
X[2]:=strtofloat(StringGrid2.cells[4,2]);
X[3]:=strtofloat(StringGrid2.cells[4,3]);
X[4]:=strtofloat(StringGrid2.cells[4,4]);
y[1]:=strtofloat(StringGrid2.cells[5,1]);
y[2]:=strtofloat(StringGrid2.cells[5,2]);
y[3]:=strtofloat(StringGrid2.cells[5,3]);
y[4]:=strtofloat(StringGrid2.cells[5,4]);
Kv_[1]:=strtofloat(StringGrid2.cells[6,1]);
Kv_[2]:=strtofloat(StringGrid2.cells[6,2]);
Kv_[3]:=strtofloat(StringGrid2.cells[6,3]);
Kv_[4]:=strtofloat(StringGrid2.cells[6,4]);
D[1]:=strtofloat(StringGrid2.cells[7,1]);
D[2]:=strtofloat(StringGrid2.cells[7,2]);
D[3]:=strtofloat(StringGrid2.cells[7,3]);
D[4]:=strtofloat(StringGrid2.cells[7,4]);
t[1]:=strtofloat(StringGrid2.cells[8,1]);
t[2]:=strtofloat(StringGrid2.cells[8,2]);
t[3]:=strtofloat(StringGrid2.cells[8,3]);
t[4]:=strtofloat(StringGrid2.cells[8,4]);
S[1]:=strtofloat(StringGrid2.cells[9,1]);
S[2]:=strtofloat(StringGrid2.cells[9,2]);
S[3]:=strtofloat(StringGrid2.cells[9,3]);
S[4]:=strtofloat(StringGrid2.cells[9,4]);
T_[1]:=strtofloat(StringGrid2.cells[10,1]);
T_[2]:=strtofloat(StringGrid2.cells[10,2]);
```

```
T_[3]:=strtofloat(StringGrid2.cells[10,3]);
T_[4]:=strtofloat(StringGrid2.cells[10,4]);
Vp[1]:=strtofloat(StringGrid2.cells[11,1]);
Vp[2]:=strtofloat(StringGrid2.cells[11,2]);
Vp[3]:=strtofloat(StringGrid2.cells[11,3]);
Vp[4]:=strtofloat(StringGrid2.cells[11,4]);

for i:=1 to 4 do begin
if i=1 then begin
Vp[1]:= KV[1]*Cv[1]*(exp(q[1]*ln(D[1])))/
((exp(m[1]*ln(T_[1])))*(exp(y[1]*ln(S[1]))));
edit1.text:=floattostr(Vp[1]);
StringGrid2.cells[11,1]:=floattostr(Vp[1]);
 end;
                          …
if i=2 then begin
Vp[2]:= KV[2]*Cv[2]*(exp(q[2]*ln(D[2])))/
((exp(m[2]*ln(T_[2])))*(exp(y[2]*ln(S[2])))*(exp(x[2]*ln(t[2]))));
edit2.text:=floattostr(Vp[2]);
StringGrid2.cells[11,2]:=floattostr(Vp[2])
End;

 if i=3 then begin
Vp[3]:=KV[3]*Cv[3]*(exp(q[3]*ln(D[3])))/
((exp(m[3]*ln(T_[3])))*(exp(y[3]*ln(S[3])))*(exp(x[3]*ln(t[3]))));
edit3.text:=floattostr(Vp[3]);
StringGrid2.cells[11,3]:=floattostr(Vp[3])
End;
                           ...
if i=4 then begin
Vp[4]:= KV[4]*Cv[4]*(exp(q[4]*ln(D[4])))/
((exp(m[4]*ln(T_[4])))*(exp(y[4]*ln(S[4])))*(exp(x[4]*ln(t[4]))));
edit4.text:=floattostr(Vp[4]);
StringGrid2.cells[11,4]:=floattostr(Vp[4])
End;
        end;


 end;
                                                …
                                          interface
```

```
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Buttons, Grids;

type
TForm1 = class(TForm)
StringGrid1: TStringGrid;
BitBtn1: TBitBtn;
Edit1: TEdit;
Edit2: TEdit;
StringGrid2: TStringGrid;
Edit3: TEdit;
Edit4: TEdit;
Edit5: TEdit;
Edit6: TEdit;
Edit7: TEdit;
Edit8: TEdit;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Edit9: TEdit;
Edit10: TEdit;
procedure BitBtn1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;
var   Cp,x,y,q,w,Kp,t,Sz,B,Z,D,n,Pz,vf,Ne,
```

```
Cv_,q_,m_,y_,u_,p_,T_,Kv_, x_, D_,t1_,Sz_, B_,Z_,Vp_,
Knv,Kmv,Kuv,np,npr,Smras,Szf,ll,pii,npac,DD  :real;
implementation

{$R *.dfm}

procedure TForm1.BitBtn1Click(Sender: TObject);
begin

Kmv:=0.51; Knv:=1; Kuv:=1;
Kp:=Kmv*Knv*Kuv;
pii:=3.14;
//---------------------------------------b
Cp:=strtofloat(StringGrid1.cells[0,1]);
x:=strtofloat(StringGrid1.cells[1,1]);
y:=strtofloat(StringGrid1.cells[2,1]);
q:=strtofloat(StringGrid1.cells[3,1]);
w:=strtofloat(StringGrid1.cells[4,1]);
Kp:=strtofloat(StringGrid1.cells[5,1]);
t:=strtofloat(StringGrid1.cells[6,1]);
Sz:=strtofloat(StringGrid1.cells[7,1]);
B:=strtofloat(StringGrid1.cells[9,1]);
D:=strtofloat(StringGrid1.cells[10,1]);
n:=strtofloat(StringGrid1.cells[11,1]);
Pz:=strtofloat(StringGrid1.cells[12,1]);
                …
 Cv_:=strtofloat(StringGrid2.cells[0,1]);
 q_:=strtofloat(StringGrid2.cells[1,1]);
 m_:=strtofloat(StringGrid2.cells[2,1]);
 y_:=strtofloat(StringGrid2.cells[3,1]);
 u_:=strtofloat(StringGrid2.cells[4,1]);
 p_:=strtofloat(StringGrid2.cells[5,1]);
 T_:=strtofloat(StringGrid2.cells[6,1]);
Kv_:=strtofloat(StringGrid2.cells[7,1]);
 x_:=strtofloat(StringGrid2.cells[8,1]);
 D_:=strtofloat(StringGrid2.cells[9,1]);
t1_:=strtofloat(StringGrid2.cells[10,1]);
Sz_:=strtofloat(StringGrid2.cells[11,1]);
 B_:=strtofloat(StringGrid2.cells[12,1]);
 Z_:=strtofloat(StringGrid2.cells[13,1]);
Vp_:=strtofloat(StringGrid2.cells[14,1]);
//  Cv_:=strtofloat(edit9.text);
```

```
…

//----------------------------------
Vp_:=Kp*(Cv_*exp(q_*ln(D_)))/
(exp(m_*ln(T_))*exp(x_*ln(t))* exp(m_*ln(Sz_))*
exp(u_*ln(B_))*exp(p_*ln(Z_)));
edit3.text:=floattostr(Vp_);
StringGrid2.cells[14,1]:=floattostr(Vp_);
npac:=1000* Vp_/(3.14*D_);
edit4.text:=floattostr(npac);
    npr:=strtofloat(edit5.text);
Vf:= (3.14*D_*npr)/(1000);
edit6.Text:=floattostr(Vf);
…
Smras:=Sz_*Z_*npr;
edit7.text:=floattostr(Smras);
Szf:= Smras/(npr*Z_);
edit8.Text:=floattostr(Szf);
…
Ne:= Pz*Vf/(1020*60);
edit2.text:=floattostr(Ne);
…
end;
…
procedure TForm1.FormCreate(Sender: TObject);
//var   Cp,x,y,q,w,Kp,t,Sz,B,Z,D,n,Pz :real;
begin
…
Kmv:=0.51; Knv:=1; Kuv:=1;
Kp:=Kmv*Knv*Kuv;
…
StringGrid1.leftcol:=13;
StringGrid1.toprow:=2;
StringGrid1.colcount:=13;
StringGrid1.rowcount:=2;
D:=80;  DD:=80;
…
StringGrid1.cells[0,0]:='Cp';  Cp:=68.2;
StringGrid1.cells[0,1]:=floattostr(Cp);
StringGrid1.cells[1,0]:='x';   x:=0.86;     StringGrid1.cells[1,1]:=floattostr(x);
…
end.
```

9. ЗАВДАННЯ НА РОЗРОБЛЕННЯ ПРОГРАМИ

Необхідно розробити програму для розрахунку режимів різання деталей авіаційного двигуна.

Для цього слід отримати у викладача завдання і матеріали для розрахунку режимів різання, указані в курсовому проекті за курсом «Проектування обробки поверхонь деталі» з дисципліни «Методи і параметри формоутворення поверхонь деталей авіаційного двинуна» [2, 3].

Наприклад, потрібно розробити програму для розрахунку режимів різання для деталі типу вал-шестерня. Є ескіз деталі із заданими табличними характеристиками поверхонь (рис. 9.1).

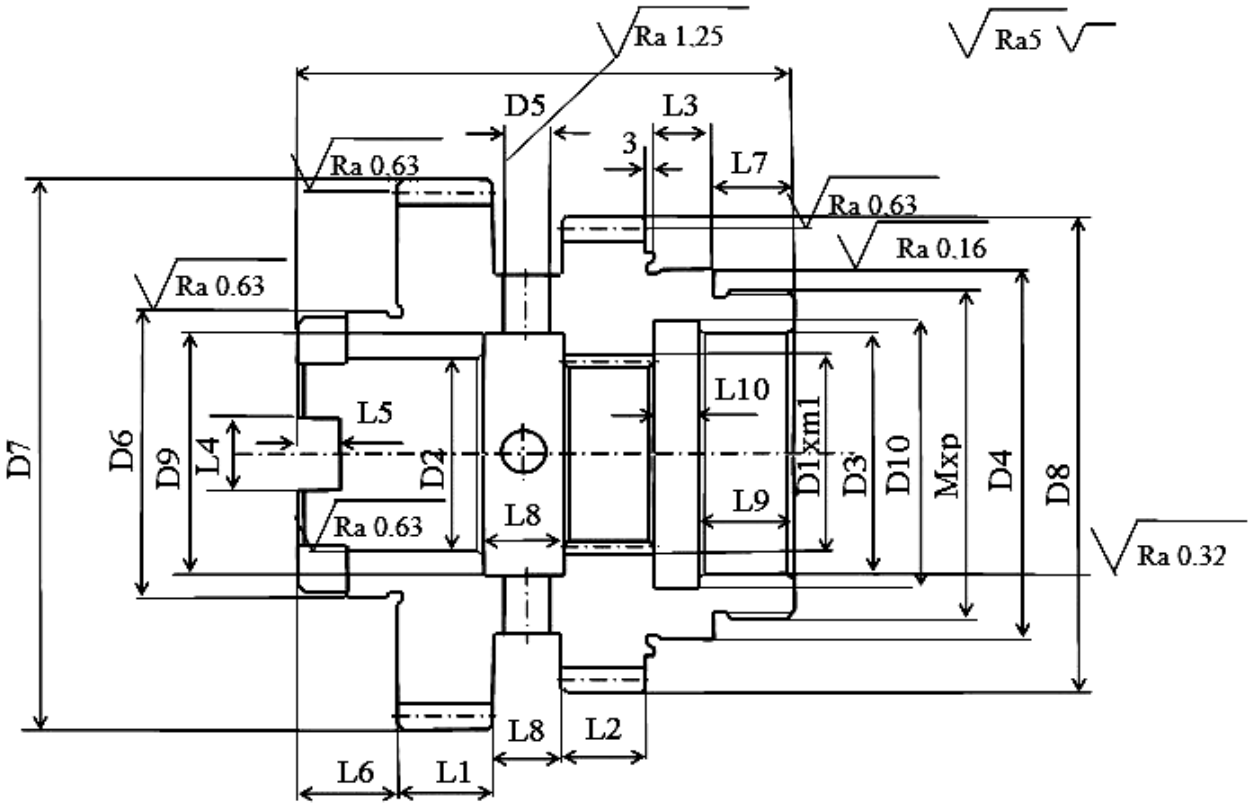

Рис. 9.1. Ескіз деталі авіаційного двигуна

Дані про розміри деталі занесено в табл. 9.1.

Таблиця 9.1

Розміри деталі, що виготовляється, мм

Параметр	Значення	Параметр	Значення
D1	28	D7	92,5
m1	1	D8	80
D2	34	D7min	81,3

Закінчення табл. 9.1

Параметр	Значення	Параметр	Значення
D3	32	L1	12
D4	46	L2	12
D5	10	L3	30
D6	48	L4	8
D9	36	L5	8
D10	34	L6	25
mA	2,5	L7	15
zA	35	L8	12
mБ	2,5	L9	18
zБ	30	L10	4
M	42	L11	109
P	2	–	–

Марка оброблюваного матеріалу – 45ХН2МФА.
План оброблення деталі містить такі етапи і операції:

Етап заготовочний:

А. Штампування.

Етап чорновий**:**

05. Фрезерно-центрувальна.
10. Токарна.
15. Токарна.
20. Токарна.
25. Токарна.
30. Токарна.

Етап термічний**:**

32. Термообробка.

Етап напівчистовий**:**

35. Токарна.
40. Токарна.
45. Токарна.
50. Токарна.
55. Токарна.

Етап чистовий:

60. Свердлувальна.
65. Фрезерна.

70. Зубофрезерна.
75. Зубодовбальна.
80. Протяжна.
85. Різенарізальна.
90. Гартувальна.

Етап обробний:

95. Круглошліфовальна.
100. Внутрішньошліфувальна.
110. Зубошліфувальна.
115. Зубошевінгувальна.
120. Хонінговальна.
125. Суперфінішна.

9.1. Алгоритми і формули для створення програми

9.1.1. Проектування токарних операцій

Оброблення точінням деталі авіаційного двигуна здійснюється в три етапи – чорновий, напівчистовий і чистовий. Точіння можна виконати на токарно-гвинторізному верстаті 16Б05П.

Подачі для кожного виду точіння визначають за такою залежністю:

– чорнове – $S=\sqrt{8 \cdot r \cdot Rz}$ = 0,89 мм/об; (9.1)

– напівчистове – $S = k_1 \cdot D_{max} + k_2 \cdot Rz + b$ = 0,67 мм/об; (9.2)

– чистове – $S= k_1 \cdot D_{max} + k_2 \cdot Rz + b$ = 0,35 мм/об, (9.3)

де коефіцієнти k_1, k_2, b вибирають з нормативних даних [3, с. 32].

Отримані значення подач порівнюють з паспортними даними верстата і призначають найближчі верстатні подачі, розкладаючи їх для вибраного верстата (табл. 9.1 – 9.9).

Таблиця 9.2

Значення верстатних подач

Вид точіння	Значення верстатної подачі, мм
Чорнове	0,82
Напівчистове	0,58
Чистове	0,34

Швидкість різання розраховують за емпіричною формулою [3, с. 30]

$$Vp = \frac{C_v}{T^m \cdot t^x \cdot S^y} K_v, \qquad (9.4)$$

де C_v – значення коефіцієнта [2, табл. 17, с. 269]; T – період стійкості різального інструменту, хв (для різців – 30–60); t – глибина різання, мм; S – подача деталі, мм/об; m, x, y – показники степеня [2, табл. 17, с. 269];

$$K_v = K_{mv} + K_{nv} + K_{uv}, \qquad (9.5)$$

де K_{mv}, K_{nv}, K_{uv} – значення коефіцієнтів [2, табл. 1 – 6, с. 261 – 263].

При чорновому точінні припуск дорівнює 2,2 мм, знімають за чотири проходи з припуском на кожні 0,55 мм.

Потім розраховують припуски на кожному проході.

Таблиця 9.3

Визначення швидкості різання

Вид точіння	S, мм/об	t, мм	T	C_v	M	x	y	K_{mv}	K_v	V_p, м/хв
Чорнове	0,82	0,55	40	350	0,2	0,15	0,35	0,51	0,129	25,23
Напівчистове	0,58	0,8	40	350	0,2	0,15	0,35	0,51	0,143	29,7
Чистове	0,34	0,6	40	350	0,2	0,15	0,35	0,51	0,143	37,3

Частоту обертання заготовки знаходять за залежностю [3, с. 31]

$$n_p = \frac{1000 \cdot V_p}{\pi \cdot D_{max}}. \qquad (9.6)$$

Таблиця 9.4

Визначення частоти обертання

Вид точіння	V_p, м/хв	n_p, мм/об
Чорнове	25,23	86,87
Напівчистове	29,7	102,25
Чистове	37,3	128,42

Отримані значення частот обертання зіставляють з верстатними даними згідно з паспортом верстата і призначають найближчі стандартні значення, розкладаючи їх для вибраного верстата.

Таблиця 9.5

Верстатні значення частот обертання

Вид точіння	Частота обертання, мм/об
Чорновое	80
Напівчистове	100
Чистове	126

Ураховуючи прийняті значення частот обертання заготовки, визначають фактичні значення швидкості різання за формулою [3, с. 31]

$$V_{ф} = \frac{\pi \cdot D_{max} \cdot n_{пр}}{1000}. \qquad (9.7)$$

Таблиця 9.6

Фактичне значення швидкості різання

Вид точіння	$V_{ф}$, м/хв
Чорнове	23,2
Напівчистове	29,045
Чистове	36,6

Основний час, який витрачається на один перехід, розраховують за залежністю [3, с. 35]

$$T_{о} = \frac{L_{р.х}}{n_{пр} \cdot S_{пр}}. \qquad (9.8)$$

Величина робочого ходу

$$L_{р.х} = l_{обр} + l_{вріз} + l_{пер} + l_{підв} = 33 \text{ мм}. \qquad (9.9)$$

Таблиця 9.7

Визначення основного часу

Вид точіння	Частота обертання, мм/об	$V_ф$, м/хв	Основний час Т, хв
Чорнове	80	23,2	0,504
Напівчистове	100	29,045	0,57
Чистове	126	36,6	0,77

Зусилля різання визначають за формулою [2, с. 271]

$$P_z = 10 \cdot C_p \cdot t^x \cdot S^y \cdot V^n \cdot K_p, \qquad (9.10)$$

де C_p – стала, x, y, n – показники степеня [2, табл. 22, с. 273].

Таблиця 9.8

Визначення зусилля різання

Вид точіння	t	S	V	C_p	x	y	n	K_p	P_z, Н
Чорнове	0,55	0,82	23,2	300	1	0,75	0,15	0,932	818,2
Напівчистове	0,8	0,58	29,045	300	1	0,75	0,15	0,932	883,87
Чистове	0,6	0,34	36,6	300	1	0,75	0,15	0,932	427,21

Потужність різання знаходять так [2, с. 271]:

$$N_e = \frac{P_Z \cdot V}{1020 \cdot 60}. \qquad (9.11)$$

Таблиця 9.9

Визначення потужності різання

Вид точіння	Значення потужності, кВт
Чорнове	0,31
Напівчистове	0,42
Чистове	0,26

9.1.2. Проектування свердлувальних операцій

Цю операцію проектують для чотирьох отворів і складається вона з таких переходів: свердлення, зенкування, чорнове і чистове розгортання.

Режими різання при свердленні розраховують для чотирьох отворів, розташованих між двома зубчастими вінцями. Розрахунок подано для одного отвору, для інших – розрахунок аналогічний.

Для розрахунків використовують табл. 9.10 – 9.18.

Подачі призначають за таблицями [2, табл. 25 – 27, с. 277).

Таблиця 9.10

Призначення подач

Вид обробки	Подача, мм/об
Свердлення	0,13
Зенкування	0,5
Розгортання чорнове	0,8
Розгортання чистове	0,56

Швидкості різання знаходять за формулами [2, с. 276]:

– для свердлення

$$V_p = \frac{C_V \cdot D^q}{T^m \cdot S^y} K_V; \qquad (9.12)$$

– для зенкування і розгортання

$$V_p = \frac{C_V \cdot D^q}{T^m \cdot t^x \cdot S^y} K_V, \qquad (9.13)$$

де C_v, q, m, x, y – значення коефіцієнтів [2, табл. 28 – 29, с. 278 – 279);

T – період стійкості [2, табл. 30, с. 279].

Таблиця 9.11

Коефіцієнти для розрахунку швидкості різання

Вид обробки	K_{mv}	K_{uv}	K_{lv}	K_v
Свердлення	0,51	0,4	1	0,204
Зенкування	0,51	1	1	0,51
Розгортання чорнове	0,51	1	1	0,51
Розгортання чистове	0,51	1	1	0,51

Таблиця 9.12

Коефіцієнти для визначення швидкості різання

Вид обробки	C_y	q	m	X	y	K_y	D, мм	t, мм	S, мм/об	T, хв	V_p, м / хв
Свердлення	7	0,4	0,2	-	0,7	0,20	8,7	4,35	0,13	35	6,94
Зенкування	18	0,6	0,2	0	0,3	0,51	9,5	0,8	0,5	1	18,2
Розгортання чорнове	101	0,3	0,4	0	0,65	0,51	9,8	0,3	0,8	20	35,1
Розгортання чистове	101	0,3	0,4	0	0,65	0,51	10	0,2	0,56	20	43,1

Таблиця 9.13

Визначення крутного моменту при свердленні і зенкуванні

Вид обробки	$C_м$	q	x	y	K_p	D, мм	t, мм	S, мм/об	$M_{кр}$, Н·м
Свердлення	0,0345	2,0	-	0,8	1,4	8,7	4,35	0,13	7,14
Зенкування	0,09	1,0	0,9	0,8	1,4	9,5	0,8	0,5	5,63

Таблиця 9.14

Визначення крутного моменту при розгортанні

Вид обробки	C_p	X	y	Z	D, мм	t, мм	S_z, мм/зуб	K_p	$M_{кр}$, Н·м
Розгортання чорнове	339	1,0	0,5	2	9,5	0,3	0,4	1,4	6,11
Розгортання чистове	339	1,0	0,5	2	10	0,2	0,28	1,4	3,58

Крутний момент визначають за залежністю [2, с. 277]:

– для свердлення

$$M_{кр} = 10 \cdot C_м \cdot D^q \cdot S^y \cdot K_P; \qquad (9.14)$$

– для зенкування

$$M_{кр} = 10 \cdot C_м \cdot D^q \cdot t^x \cdot S^y \cdot K_P; \qquad (9.15)$$

– для розгортання

$$M_{кр} = \frac{C_P \cdot t^x \cdot S_z^y \cdot D \cdot z}{2 \cdot 100}, \qquad (9.16)$$

де C_M, C_P, q, x, y – значення коефіцієнтів [2, табл. 32, с. 281].

Осьову силу розраховують за формулами [2, с. 277]:

– для свердлення

$$P_o = 10 \cdot C_P \cdot D^q \cdot S^y \cdot K_P, \qquad (9.17)$$

– для зенкування

$$P_o = 10 \cdot C_P \cdot t^x \cdot S^y \cdot K_P, \qquad (9.18)$$

де C_P, q, x, y – значенння коефіцієнтів [2, табл. 32, с. 281].

Таблиця 9.15

Визначення осьових сил

Вид обробки	C_P	q	x	y	K_p	D, мм	t, мм	S, мм/об	P_o, Н
Свердлення	68	1,0	-	0,7	1,4	8,7	4,35	0,13	1985,7
Зенкування	67	-	1,2	0,65	1,4	9,5	0,8	0,5	462,25

Частоту обертання визначають за формулою [2, с. 280]

$$n_p = \frac{1000 \cdot V_p}{\pi \cdot D_{max}} \qquad (9.19)$$

і вибирають найближче верстатне значення. Верстат – вертикально-свердлувальний 2Н125.

Таблиця 9.16

Визначення частот обертання

Вид обробки	V, м/хв	D, мм	n, об/хв	$n_{пр}$, об/хв
Свердлення	6,94	8,7	254,04	186,7
Зенкування	18,2	9,5	610,12	482,06
Розгортання чорнове	35,12	9,8	1141,29	774,6
Розгортання чистове	43,07	10	1371,6	1244,7

Фактичні значення швидкостей різання обчислюють за формулою

$$V_{ф} = \frac{\pi \cdot D_{max} \cdot n_{пр}}{1000}. \quad (9.20)$$

Таблиця 9.17

Фактичні значення швидкостей різання

Вид обробки	D, мм	$n_{пр}$, об/хв	$V_{ф}$, м/хв
Свердлення	8,7	186,7	5,1
Зенкування	9,5	482,06	14,4
Розгортання чорнове	9,8	774,6	23,8
Разгортання чистове	10	1244,7	39,1

Потужність різання визначають за такою залежністю [2, с. 280]:

$$N_e = \frac{M_{КР} \cdot n}{9750}. \quad (9.21)$$

Таблиця 9.18

Розрахунок потужності різання

Вид обробки	$M_{КР}$, Н·м	$n_{пр}$, об/хв	N_e, кВт
Свердлення	7,14	186,7	0,13
Зенкування	5,63	482,06	0,28
Розгортання чорнове	6,11	774,6	0,49
Розгортання чистове	3,58	1244,7	0,46

9.1.3. Проектування операції фрезерування

Цю операцію застосовують для отримання пазів на деталі. Для розрахунків використовують табл. 9.19 – 9.20. Фрезерний верстат для оброблення – 6Р10. Інструмент – фреза дискова пазова за ГОСТ 1695-80 діаметром 80 мм (матеріал – Т15К6).

Геометричні параметри фрезерування:

– глибина фрезерування t = 8 мм;

– ширина фрезерування B = 8 мм.

Значення поправкових коефіцієнтів:

$$K_p = K_{mv} \cdot K_{nv} \cdot K_{uv};\ K_{mv} = 0{,}51;\ K_{nv} = 1;\ K_{uv} = 1.$$

Швидкість різання при фрезеруванні визначають за формулою [2, с. 282]

$$V_P = \frac{C_V \cdot D^q}{T^m \cdot t^x \cdot S_z^{\ m} \cdot B^u \cdot Z^p} \cdot K_P, \qquad (9.22)$$

де C_V, q, x, y, m, u, p – значення коефіцієнтів [2, табл. 39, с. 286].

Таблиця 9.19

Визначення швидкості різання

C_V	q	m	y	u	p	T	K_v	x
75,5	0,25	0,2	0,2	0,1	0,1	120	0,51	0,2
D, мм	t, мм		S_Z, мм/зуб		B, мм	Z	V_P, м/хв	
80	8		0,1		12	18	29,2	

Розрахункову частоту знаходять за залежністю [3, с. 42]

$$n_p = \frac{1000 \cdot Vp}{\pi \cdot D} = \frac{1000 \cdot 29{,}2}{\pi \cdot 80} = 116{,}24 \text{ об/хв.} \qquad (9.23)$$

За паспортом верстата вибирають стандартну частоту обертання n = 99,6 об/хв.

Фактичне значення швидкості різання [3, с. 42]

$$V_ф = \frac{\pi \cdot D \cdot n_{пр}}{1000} = \frac{\pi \cdot 80 \cdot 99{,}6}{1000} = 25{,}02 \text{ м/хв.} \qquad (9.24)$$

Хвилинна подача [3, с. 43]

$$S_{хв}^{розр} = S_z \cdot Z \cdot n_{пр} = 0{,}1 \cdot 18 \cdot 99{,}6 = 179{,}28 \text{ мм/хв.} \qquad (9.25)$$

Фактичне значення подачі на зуб [3, с. 43]

$$S_z^ф = \frac{S_{хв}^{розр}}{n_{пр} \cdot Z} = \frac{179{,}28}{99{,}6 \cdot 18} = 0{,}1 \text{ мм/зуб.} \qquad (9.26)$$

Колову силу розраховують за формулою [3, с. 43]

$$P_Z = \frac{10 \cdot C_P \cdot t^x \cdot S_Z^y \cdot B \cdot Z}{D^q \cdot n^w} \cdot K_P, \qquad (9.27)$$

де C_p , q, x, y, w – значення коефіцієнтів [2, табл. 41, с. 291].

Таблиця 9.20

Визначення сили різання

C_p	x	y	q	w	K_P	t	S_z	B	Z	D	n	P_Z, Н
68,2	0,86	0,72	0,86	0	1,1	8	0,1	8	18	80	99,6	708,67

Ефективну потужність визначають за формулою [3, с. 43]

$$N_e = \frac{P_Z \cdot V_ф}{1020 \cdot 60} = \frac{708{,}67 \cdot 25{,}02}{1020 \cdot 60} = 0{,}29 \text{ кВт}. \qquad (9.28)$$

9.1.4. Проектування зубофрезерної операції

Цю операцію застосовують для отримання зубчастої поверхні на одному з вінців.

Для розрахунків використовують табл. 9.21 – 9.24.

Параметри оброблюваної зубчастої поверхні: m = 2,5; z = 35; B = 12; НВ = 340. Вибір різального інструменту: фреза черв'ячна модульна m = 2,5; d_a = 71; d = 27; d_1 = 40; L = 63; ГОСТ 9324-80; матеріал фрези - Р6М5.

Вибір устаткування: верстат зубофрезерный 5А326; потужність електродвигуна приводу головного руху – 7,5 кВт; група устаткування – III [5, табл. 40, с.147].

Визначення подачі: табличне значення – 4 мм/об [5, табл. 41, с. 149]; поправкові коефіцієнти [5, табл. 42, с.150]

Таблиця 9.21

Визначення коефіцієнта K_S

НВ	До 220	До 300
K_S	1	0,7

У нашому випадку K_S = 0,7 НВ.

Таблиця 9.22

Визначення коефіцієнта K_{S1}

В	0	30	45
K_{S1}	1	0,8	0,65

У нашому випадку K_{S1} = 1.

Таблиця 9.23

Визначення коефіцієнта K_{S2}

$K_{зах}$	1	2	3
K_{S2}	1	0,7	0,5

У нашому випадку K_{S2} = 1.

Тоді значення подачі

$$S_p = S_т \cdot K_S = 4 \cdot 0,7 = 2,8 \text{ мм/об.}$$

Після цього знаходять стійкість фрези Т (див. табл. 9.24).

Таблиця 9.24

Визначення стійкості черв'ячної фрези [6, табл. 37, с.142]

Модуль m	До 4	До 6	До 8	До 12	Понад 12
Значення	4	6	8	12	16

Т = 4 ч = 240 хв, оскільки заданий модуль m менше 4.

Визначення швидкості різання [5, табл. 45 – 51, с. 154 – 160] v =14 м/хв. За номограмою швидкостей встановлюють кількість оборотів [5, дод. 3] n = 59 об/хв. Настроювання гітари розподілу [5, дод. 4] z = 35, a = 40, b = c = 50,d = 70. Настроювання вертикальної подачі фрезерного супорта S = 2,8 мм/об, $a_1/b_1 \cdot c_1/d_1$ = 40/41 · 47/59 = 0,8.

Встановлюють глибину фрезерування.

Колесо нарізають за один прохід.

Глибина фрезерування h_1 = 2,2 · m = 2,2 · 2,5 = 5,5 мм.

Довжина робочого ходу

$$L_{р.х} = l_{вріз} + l_{обр} + l_{пер}, \quad (9.29)$$

де $l_{вріз}$ = 19 мм (з геометричних міркувань);

$l_{обр}$ = B = 12 мм;

$l_{пер}$ = 1 мм.

Разом $L_{р.х}$ = 32 мм.

Основний робочий час

$$T_o = \frac{L_{р.х} \cdot Z}{S_B \cdot n} = \frac{32 \cdot 35}{2{,}8 \cdot 59} = 6{,}8 \text{ хв.} \quad (9.30)$$

9.1.5. Проектування зубодовбальної операції

Параметри нарізуваного колеса – z = 30; m = 2,5; B = 12; D = 80.
Параметри довбяка: тип 1; клас А ГОСТ 9323-79; m = 2,5;
z_0 = 40; d_0 = 100; d_{a0} = 107,75; B = 20; матеріал – Р6М5.

Вибір устаткування – зубодовбальний верстат 5122.

Для розрахунків використовують табл. 9.25 – 9.30.

Таблиця 9.25

Визначення колової подачі [5, табл. 28, с. 143]

Вид обробки	m	$S_{кр.п}$
Чорнова	До 2 2 … 4 до 5	0,45 0,35 … 0,45 0,2 … 0,4
Чистова	2 …10	0,25 ... 0,3

У нашому випадку $S_{кр.п}$ = 0,4.

Таблиця 9.26

Визначення коефіцієнта K_S

$S_{кр.п}$	1	0,9	0,8	0,7
НВ	до 190	190…220	220…240	240…300

У нашому випадку $S_{кр.п}$ = 0,7.

Колова розрахункова подача

$$S_{кол.р} = S_{кол.\ п} \cdot K_s; \quad (9.31)$$

$$S_{кол.р} = 0,4 \cdot 0,7 = 0,28 \text{ мм/подв.х.}$$

Радіальна подача

$$S_p = (0,1…0,3) \cdot S_{кр} \text{ [5, с.139].} \quad (9.32)$$

У нашому випадку $S_p = 0,2 \cdot 0,28 = 0,056$ мм/подв.х.

Таблиця 9.27

Визначення швидкості різання [5, табл. 29, с. 144]

$S_{кр}$	m			
	До 2	До 4	4.6	6.12
0,1	41	33	26	25
0,13	36	23	24	22
0,16	32	26	22	20
0,2	29	23	20	18
0,26	25	21	17	16
0,32	23	18	15	14
0,42	20	16	14	13
0,52	18	14	12	11

За табл. 26 вибирають швидкість різання V = 21 м/хв.

Коефіцієнт K_{v1}, який ураховує механічні властивості, визначають за табл. 9.28. Якщо границя міцності $\sigma_в$ = 880 і твердість 340 НВ, то вибирають K_{v1} = 0,4. Коефіцієнт K_{v2} ураховує вид оброблюваного матеріалу (табл. 9.29).

Таблиця 9.28

Визначення коефіцієнта Kv1 [5, табл. 13, с. 126]

НВ	$\sigma_в$	K_{v1}
180	550 … 600	1,25
190	650 ... 700	1
220	750 ... 800	0,8
250	850 ... 900	0,4

Таблиця 9.29

Визначення коефіцієнта K_{v2}

Тип сталі	K_{v2}
Вуглецева сталь 35…45	1
Низьколегована сталь 40Х, 45ХН	0,9
Легована сталь 18Х2Н4ВА	0,75

Коефіцієнт K_{v3} залежить від виду обробки і вибирають його за табл. 9.30.

Таблиця 9.30

Визначення коефіцієнта K_{v3}

Вид обробки	K_{v3}
Чорнова	1
Напівчистова	1,5
Чистова	1,2

У нашому випадку K_{v3} = 1.

Тоді швидкість різання

$$V = 21 \cdot 0{,}4 \cdot 0{,}75 \cdot 1 = 6{,}3 \text{ м/хв.} \quad (9.33)$$

Кількість подвійних ходів визначають за формулою

$$n_{\text{подв.х}} = \frac{1000 \cdot V}{2 \cdot L_{\text{р.х}}} = \frac{1000 \cdot 6{,}3}{2 \cdot 16} = 196{,}875 \text{ подв.х/хв ;} \quad (9.34)$$

$$L_{\text{р.х}} = l_{\text{підх}} + B + l_{\text{пер}} = 16 \text{ мм;}$$

$n_{\text{подв.х}} = 197$; вважаємо $n_{\text{подв.х}} = 200$ дв.х/хв.

Визначення основного часу:

$$T_o = \frac{\pi \cdot m \cdot z}{S_{\text{кр}} \cdot n_{\text{подв.х}}} + \frac{h}{S_{\text{рад}} \cdot n_{\text{подв.х}}}; \quad (9.35)$$

$h = 2{,}2 \cdot m = 5{,}5$; $S_{\text{кр}} = 0{,}26$ мм/дв.х; $S_{\text{рад}} = 0{,}056$ мм/дв.х;

$$T_o = \frac{\pi \cdot m \cdot z}{S_{\text{кр}} \cdot n_{\text{подв.х}}} + \frac{h}{S_{\text{рад}} \cdot n_{\text{подв.х}}} = \frac{\pi \cdot 2{,}5 \cdot 30}{0{,}26 \cdot 200} + \frac{5.5}{0{,}056 \cdot 200} = 5 \text{ хв.}$$

Визначення потрібної потужності різання:

$N_e = N_{\text{табл}} \cdot K_{2N}$; $N_{\text{табл}} = 1{,}1$ кВт; $K_{2N} = 0{,}75$.

$$N_e = N_{\text{табл}} \cdot K_{2N} = 1{,}1 \cdot 0{,}75 = 0{,}825 \text{ кВт;}$$

КПД верстата $\eta = 0{,}6$.

Потужність верстата $N = N_{\text{ст}} \cdot \eta = 3 \cdot 0{,}6 = 1{,}8$ кВт. (9.36)

9.1.6. Проектування протяжної операції

Таку операцію використовують для отримання в отворі шліцьової поверхні. Параметри притягуваного контуру: внутрішні шліци; модуль m = 1; кількість зубів – 26; ділильний діаметр – 26 мм; діаметр вершин – 25 мм; діаметр западин – 27 мм; довжина обробки – 38 мм.

Вибирають протяжку для шліцьових отворів евольвентним профілем за ГОСТ 25157-82.

Група швидкості різання – IV [2, табл. 53, с. 299], оскільки НВ = 340.

Визначення допустимої швидкості протягування V = 2,5 м/хв [2, табл. 52, с. 299].

Схема протягування – генераторна (рис. 9.2).

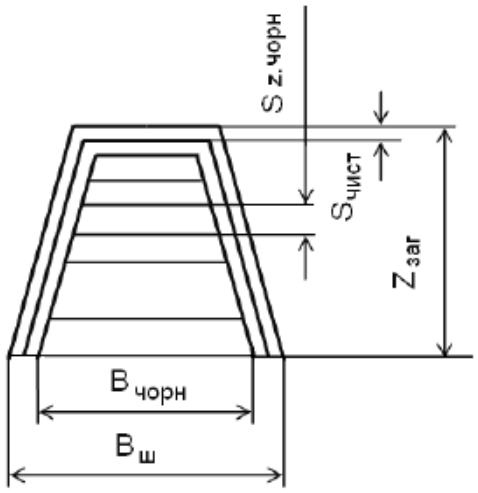

Рис. 9.2. Генераторна схема протягування

Геометричні параметри протяжки [2, табл. 53, с. 299]:

– крок зубів протяжки $t = 1{,}5 \cdot \sqrt{l_д} = 9{,}25$ мм; (9.37)

$$\frac{l_д}{t} \geq 3 = 4{,}1 \text{ мм};$$

– робоча висота профілю $h_p = 0{,}4 \cdot t = 3{,}7$ мм; (9.38)

– радіус закруглення $r = 0{,}5 \cdot h_p = 1{,}85$ мм. (9.39)

Зіставлення отриманих значень з табличними [2, табл. 62, с. 171]:

t = 9 мм; h_p = 3,6 мм; r = 1,8 мм; b = 3,5 мм; r_1 = 55 мм.

Задання кроку для чистових і калібрувальних зубів [2, табл. 63, с. 173]:

t_1 = 6 мм; t_2 = 6,5 мм; t_3 = t_1 +1 = 7 мм.

Вибір подач [2, табл. 64, с.173]:

$S_{zчорн}$ = 0,1 мм/зуб, $S_{zчист}$ = 0,06 мм/зуб.

Розділення припуску на чорновій і чистовій операціях:

$$
\begin{aligned}
Z_{заг} &= Z_{чорн} + Z_{чист} = 2{,}25\ мм;\\
Z_{чорн} &= (0{,}8...0{,}9)\cdot Z_{заг} = 1{,}8\ мм;\\
Z_{чист} &= (0{,}2...0{,}1)\cdot Z_{заг} = 0{,}45\ мм.
\end{aligned}
\qquad (9.40)
$$

Визначення кількості поясів зубів чорнової, чистової і калібрувальної частин протяжки

$$
\begin{aligned}
n_{калібр} &= 4{,}5;\\
n_{чорн} &= \frac{Z_{чорн}}{S_{чорн}} = \frac{1{,}8}{0{,}1} = 18;\\
n_{чист} &= \frac{Z_{чист}}{S_{чист}} = \frac{0{,}45}{0{,}06} = 7{,}5.
\end{aligned}
\qquad (9.41)
$$

Робоча довжина протяжки:

$$
\begin{aligned}
l_{чорн} &= t\cdot n_{чорн} = 162\ мм;\\
l_{чист} &= t_{чист}\cdot n_{чист} = 45\ мм;\\
l_{калібр} &= t_{чист}\cdot n_{калібр} = 26\ мм;\\
l_{р.ч} &= l_{чорн} + l_{чист} + l_{калібр} = 233\ мм.
\end{aligned}
\qquad (9.42)
$$

Периметр оброблюваного профілю В визначають виходячи з геометричних міркувань (рис. 9.3):

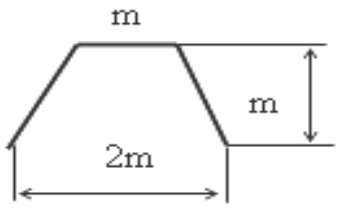

Рис. 9.3. Периметр зуба протяжки

$$
\begin{aligned}
B_{чорн} &= 0{,}8\cdot B_{ш} = 1{,}6\ мм;\\
B_{ш} &= 2\cdot m = 2\ мм;\\
B_{чист} &= 2\cdot\sqrt{(0{,}5\cdot m)^2 + m^2} + m = 3{,}232\ мм.
\end{aligned}
\qquad (9.43)
$$

$$B_{\Sigma чорн} = B_{чорн} \cdot z \cdot \frac{l_д}{t} = 175,6 \text{ мм;} \quad (9.44)$$

$$B_{\Sigma чист} = B_{чист} \cdot z \cdot \frac{l_д}{t_{чист}} = 531,87 \text{ мм.} \quad (9.45)$$

Питома сила різання на 1 мм периметра $P_{чорн}$, $P_{чист}$ [2, табл. 54, с. 300]:

$$P_{чорн} = 390 \text{ Н/мм}, \quad P_{чист} = 282 \text{ Н/мм.}$$

Зусилля різання

$$P_{\Sigma чорн} = P_{чорн} \cdot B_{\Sigma чорн} = 68484 Н;$$
$$P_{\Sigma чист} = P_{чист} \cdot B_{\Sigma чист} = 149987 Н. \quad (9.46)$$

Ефективна потужність різання

$$N_e = \frac{P_{\Sigma н.б} \cdot V}{61200 \cdot \eta} = 12,74 кВт, \quad (9.47)$$

де η=0,7.

Основний технологічний час

$$L_{р.х} = l_{р.ч} + l_{п.н.ч} + l_{з.н.ч} = 258,5 \text{ мм;} \quad (9.48)$$

$$K_{звор} = 1,2...1,5;$$

$$T_o = \frac{L_{р.х} K_{звор}}{V} = \frac{258,5 \cdot 1,2}{1000 \cdot 2,5} = 0,12 \text{ хв.} \quad (9.49)$$

Устаткування – горизонтально-протяжний автомат для внутрішнього протягування ТБ56.

9.1.7. Проектування операції різенарізання

Цю операцію використовують для нарізування різі на деталі. Параметри різі – різь метрична М42 х 2, $l_{дет}$ = 15 мм. Призначення інструмента [2, с. 122] – токарний нарізний різець з пластинами з твердого сплаву Т15К6 за ГОСТ 18885-73. Оброблення – одиничним різцем; схема

різання – профільна. Призначення устаткування – верстат токарно-гвинторізний 16Б05П.

Для розрахунків використовують табл. 9.31– 9.32.

Кількість робочих ходів – $i_{чорн}$, $i_{чист}$, i [2, табл. 45, с. 294]:

$$i_{чорн} = 3;\ \ i_{чист} = 2;\ \ i = 5.$$

Визначення подач

$$S_{под} = p = 2 \text{ мм/об;} \qquad (9.50)$$

$$S_{попер} = \frac{p \cdot \sqrt{3}}{i} = 0{,}69 \text{ мм/об.} \qquad (9.51)$$

Визначення поправкового коефіцієнта

$$K_v = K_{mv} \cdot K_{uv} \cdot K_{ov} = 0{,}51 \cdot 1{,}16 \cdot 0{,}75 = 0{,}44.$$

Швидкість різання при нарізуванні різі [2, с. 295]

$$V = \frac{C_v \cdot i^x}{T^m \cdot S^y} \cdot K_v, \qquad (9.52)$$

де C_v, q, x, y, m – значення коефіцієнтів [2, табл. 49, с. 296].

Таблиця 9.31

Швидкість різання при нарізуванні різі

C_v	x	y	m	T	i	S, мм/об	K_v	V, м/хв
244	0,23	0,3	0,2	70	5	0,69	0,44	74,4

Частота обертання шпинделя – розрахункова n_p взята з роботи [2, с. 295]:

$$n_p = \frac{1000 \cdot V_p}{\pi \cdot D_{max}} = 564{,}15 \text{ об/хв,} \qquad (9.53)$$

$$n_{пр} = 504 \text{ об/хв.}$$

Величина робочого ходу

$$L_{р.х} = l_{вріз} + l_{обр} + l_{підв} + l_{пер} = 20 \text{ хв.} \qquad (9.54)$$

Основний час

$$T_o = \frac{L_{p.x} \cdot i}{S_{прод} \cdot n_{пр}} = \frac{20 \cdot 5}{2 \cdot 504} = 0{,}0992 \text{ хв.} \qquad (9.55)$$

Сила різання

$$P_z = \frac{10 \cdot C_p \cdot p^y}{i^n} \cdot K_p, \qquad (9.56)$$

де коефіцієнти K_p вибирають з довідкових даних [2, табл. 1, 2, с. 261], C_p, y, n – [2, табл. 51, с. 298].

Таблиця 9.32

Визначення сили різання

C_p	У	n	K_p	p	i	P_z, Н
148	1,7	0,71	0,75	2	5	1536,26

Потужність верстата

$$N = \frac{P_z \cdot V}{1020 \cdot 60} = \frac{1536{,}26 \cdot 74{,}4}{1020 \cdot 60} = 1{,}86 \text{ кВт.}$$

9.1.8. Проектування круглошліфувальних операцій

Шліфування використовують для отримання більш точних параметрів поверхонь, зокрема для зниження шорсткості.

Вибір шліфувального круга

Матеріал круга – нормальний електрокорунд 15А. Застосовується для інструментів на керамічній зв'язці для оброблення сталей [1, с. 242 – 243]. Зернистість – 40-Н для чистового шліфування сталей [1, с. 245]. Зв'язка – керамічна К5. Використовується для всіх типів шліфування, окрім прорізних і відрізних робіт для інструменту підвищеної точності [1, с. 247]. Твердість – С2 для остаточного шліфування зовнішніх та інших поверхонь [1, с. 248 – 249]. Номер структури – 6. Круги зовнішнього шліфування з периферією

круга [1, с. 249]. Клас точності – А. Круги з індексом зернистості В, П, Н більш точні, ніж круги класу В [1, с. 250]. Клас незрівноваженості – 2 [1, с. 250]. Профіль – ПП [1, с. 252]. Розміри круга [1, с. 253]: D = 500 мм; d = 200 мм; Н = 50 мм. Колова швидкість заготовки [3, с. 22]

$$V_З = 13 \cdot \ln d_З = 13 \cdot \ln 48 = 21{,}85 \text{ м/хв.} \quad (9.57)$$

Частота обертання заготовки [3, с. 22]

$$n_З = \frac{1000 \cdot V_З}{\pi \cdot D_З} = \frac{1000 \cdot 21{,}85}{\pi \cdot 48} = 144{,}97 \text{об/хв;} \quad (9.58)$$

$$V_З^д = \frac{\pi \cdot D_З \cdot n_{пр}}{1000} = 22{,}61 \text{ м/хв.} \quad (9.59)$$

Верстатна частота обертання [3, с. 18]

$$n_{пр} = 150 \text{ об/хв.} \quad (9.60)$$

Частота круга $n_{кр}$ = 1100 об/хв [3, с. 18].

Колова швидкість круга [3, с. 22]

$$V_{кр}^д = \frac{\pi \cdot D_{кр} \cdot n_{кр}}{60 \cdot 1000} = 22{,}78 \text{ м/с.} \quad (9.61)$$

Радіальна подача [3, с. 22]

$$S_{рад}^{0\text{-}1} = 0{,}5 \cdot \frac{13{,}5}{d_З^{0{,}7}} = 0{,}5 \cdot \frac{13{,}5}{48^{0{,}7}} = 0{,}449 \text{ мм/хв;} \quad (9.62)$$

$$S_{рад}^{1\text{-}2} = 0{,}5 \cdot \frac{13{,}5}{d_З^{0{,}5} \cdot l_{акт}^{0{,}6}} = 0{,}5 \cdot \frac{13{,}5}{48^{0{,}5} \cdot 25^{0{,}6}} = 0{,}141 \text{ мм/хв.} \quad (9.63)$$

Середня радіальна подача [3, с. 22]

$$S_{рад}^{сер} = \frac{S_{рад}^{0\text{-}1} + S_{рад}^{1\text{-}2}}{2} = 0{,}29 \text{ мм/хв.} \quad (9.64)$$

Основний час [3, с. 22]

$$T_О = \frac{Z_1}{S_{рад}^{0\text{-}1}} + \frac{Z_2}{S_{рад}^{1\text{-}2}} = \frac{0{,}3}{0{,}449} + \frac{0{,}1}{0{,}141} = 1{,}378 \text{ хв.} \quad (9.65)$$

Ефективна потужність [3, с. 22]

$$N_{еф} = 0{,}0065 \cdot S_{рад}^{0,7} \cdot d_З^{0,7} \cdot l_{акт} = 1{,}04 \text{ кВт}. \quad (9.66)$$

9.1.9. Проектування операції зубошліфування

Операцію зубошліфування використовують для зниження шорсткості на зубчастих поверхнях деталі.

Параметри зубчастого вінця – m = 2,5; z = =35; B = 12 мм.

Механічні параметри заготовки – НВ = 480.

Підбір шліфувального круга

Матеріал круга– нормальний електрокорунд 25А.

Застосовується для інструментів на керамічній зв'язці для оброблення сталей [1, с. 242 – 243]. Зернистість – 20-Н (шліфування сталей) [1, с. 245]. Зв'язка – керамічна К8.

Використовують для всіх типів шліфування, окрім прорізних і відрізних робіт для інструменту підвищеної точності [10, с. 247].

Твердість – СМ2 для остаточного шліфування зовнішніх та інших поверхонь [10, с. 248 – 249]. Номер структури – 3. Круги зовнішнього шліфування периферією круга [10, с. 249].

Клас точності – А. Круги з індексом зернистості В, П, Н більш точні, ніж круги класу В [1, с. 250]. Клас незрівноваженості – 2 [1, с. 250].

Профіль – ПП [10, с. 252]. Значення подачі вибирають за табл. 9.33.

Розміри круга [10, с. 253]: D = 400 мм; d = 127 мм; b = 40 мм.

Призначення припуску на шліфування:

$$\begin{aligned} \nabla h_{заг} &= 0{,}1 \cdot m = 0{,}25 \text{ мм}; \\ \nabla h_{чорн} &= 0{,}8 \cdot \nabla h_{заг} = 0{,}2 \text{ мм}; \\ \nabla h_{чист} &= 0{,}2 \cdot \nabla h_{заг} = 0{,}05 \text{ мм}. \end{aligned} \quad (9.67)$$

Таблиця 9.33

Вибір значення подачі [5, с. 204]

Подача	$S_{рад}$, мм/х	$S_{верт}$, мм/об.к
Чорнова	0,06.0,08	1,4…1,8
Чистова	0,01	0,3…0,5

Кількість робочих ходів

$$i_{чорн} = \frac{\Delta h_{чорн}}{2 \cdot S_{рад.чорн} \cdot tg20°} = 3{,}9 \approx 4; \quad (9.68)$$

$$i_{чист} = \frac{\Delta h_{чист}}{2 \cdot S_{рад.чист} \cdot tg20°} = 6{,}87 \approx 7. \quad (9.69)$$

Довжина робочого ходу

$$L_{р.х} = B + (2.5) = 15 \text{ мм}. \quad (9.70)$$

Основний час оброблення

$$T_o = \frac{L_{р.х}}{n_к}\left(\frac{i_{чорн}}{S_{в.чорн}} + \frac{i_{чист}}{S_{в.чист}}\right) = 7{,}07 \text{ хв}. \quad (9.71)$$

$$n_к = \frac{n_{кр}}{z_к} = 42{,}78 \text{ хв}. \quad (9.72)$$

9.1.10. Проектування операції зубошевінгування

Цю операцію застосовують для зниження шорсткості і поліпшення якості поверхні зубів. Параметри оброблюваного зубчастого вінця: m = 2,5, z = 30, B = 12, марка матеріалу оброблюваного колеса – НВ = 480.

Шевер: m = 2,5; $z_Ш$ = 91; β = 5; d_{ao} = 234,56 мм; матеріал шевера – Р6М5, верстат – 5702 В. Метод шевінгування – паралельний.

Припуск на шевінгування $\Delta h = 0{,}035 \cdot m$ = 0,0875 мм [7, с. 205].

Вибір подачі [7, с. 205]: $S_{рад}$ = 0,04 … 0,05 мм/х (7-й ступінь точності); $S_{рад}$ = 0,02 … 0,025 мм/х (6-й ступінь точності); $S_{позд}$ = = 0,25 мм/об (табл. 9.34).

Таблиця 9.34

Вибір поздовжньої подачі

R_a	z	$S_{позд}$
1,25 …0,63	17…25	0,2 …0,3
	40 …100	0,35...0,6
0,63 ...0,32	17…25	0,15…0,25
	40...100	0,25…0,4

Швидкість різання

$$V = 90 \ldots 145 \text{ м/хв [7, с. 205].}$$

Оптимальна швидкість за стійкістю – 120 м/хв.

Частота обертання шевера

$$n_{ш} = \frac{1000 \cdot V}{\pi \cdot d} = 162{,}92 \text{ об/хв.} \quad (9.73)$$

Частота за паспортом верстата – $n_{пр}$ = 160 об/хв.

Кількість робочих ходів

$$n_{р.х} = \frac{\Delta h}{2 \cdot S_{р} \cdot tg20°} = 3. \quad (9.74)$$

Кількість зачищувальних (калібрувальних) ходів

$$n_{р.х} = 3.$$

Загальна кількість ходів

$$i = n_{р.х} + n_{к.х} = 6. \quad (9.75)$$

Хвилинна подача

$$S_{пр.м} = S_{пр} \cdot \frac{n_{ш} \cdot z_{ш}}{z_{к}} = 121{,}33 \text{ мм/хв.} \quad (9.76)$$

Величина робочого ходу

$$L_{р.х} = B + (1.3) = 15 \text{ мм.} \quad (9.77)$$

Основний час

$$T_{о} = \frac{L_{р.х} \cdot i}{S_{пр.м}} = 0{,}74 \text{ хв.} \quad (9.78)$$

9.1.11. Проектування операції хонінгування

Цю операцію використовують для доведення і поліпшення якості внутрішніх поверхонь.

Параметри оброблюваної поверхні для попереднього і поточного ступенів оброблення: поверхня – отвір ∅34Н6; L = 18 мм; R_a 0,32; попередня операція – шліфування Н7; R_a 0,63.

Визначення припуску на хонінгування [4, табл. 9, с. 29]:

$$Z = 0{,}05 \text{ мм}; \quad Z_{\text{чорн}} = 0{,}025 \text{ мм}; \quad Z_{\text{чист}} = 0{,}025 \text{ мм}.$$

Вибір устаткування для операції хонінгування – верстат 3А83 [2, табл. 23, с. 38];

Призначення інструменту

Склад брусків у хонінгувальній головці – 63С, М40, С1; бакелітова зв'язка [9, табл. 73, с. 432]. Довжину робочого ходу визначають з геометричних міркувань (рис. 9.4):

$$\begin{aligned} l &= (0{,}5...0{,}75)\cdot L = 9 \text{ мм}; \\ L_{\text{р.х}} &= L + 2\cdot l_{\text{вих}} - l = 3 \text{ мм}; \\ l_{\text{вих}} &= 3 \text{ мм}. \end{aligned} \qquad (9.79)$$

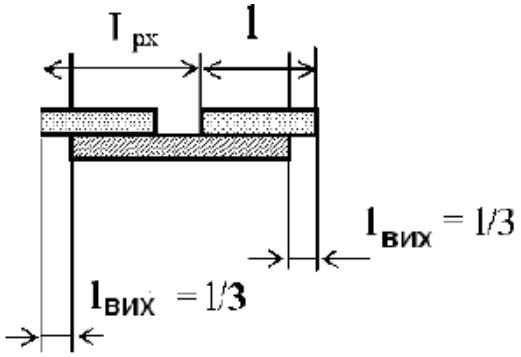

Рис. 9.4. Схема хонінгування

Різальний периметр

$$B_{\text{різ}} = B_{\text{обр}} \cdot 0{,}5 = \pi \cdot d \cdot 0{,}5;$$

$$B_{\text{різ}} = \pi \cdot 34 \cdot 0{,}5 = 54.$$

Кількість брусків – 6, ширина бруска – 9 мм.

Призначення поздовжньої подачі: при $L_{\text{р.х}} < 50$ $V_{\text{п.п}}$ = 5,8 м/хв [9, с. 435], $V_{\text{п.п}}$ = 5 м/хв.

Призначення співвідношення для колової швидкості обертання і швидкості прямолінійного руху:

$$K = \frac{V_{об}}{V_{пр}};$$

$$K_{чорн} = 3;\ K_{чист} = 7 \text{ [9, табл. 74, с. 436];}$$

$$V_{в.чорн} = 7 \cdot 2 = 14 \text{ м/хв};\ V_{в.чист} = 5 \cdot 7 = 35 \text{ м/хв.}$$

Кількість подвійних ходів

$$n_{подв.х} = \frac{V_{пр} \cdot 1000}{2 \cdot L_{р.х}}, \qquad (9.80)$$

$$n_{подв.х} = 5 \cdot 1000 / (2 \cdot 3) = 833{,}3 \text{ подв.х/хв.}$$

9.1.12. Проектування операції суперфінішування

Параметри оброблюваної поверхні – d = 46 мм, l = 30 мм.

Попередня операція – чистове шліфування, IT6, R_a 0,32.

Поточна операція – суперфінішування, IT5, R_a = 0,16.

Визначення припуску на сторону

Припуск має перевищувати висоту нерівностей на попередньому обробленні.

Для тонкого шліфування R_z = 0,5 … 8 мкм.

Призначаємо Z =15 мкм на сторону [9, табл. 75, с. 438], на розмір (діаметр) Z = 0,03 мм.

Вибір різального інструменту, розмірів і кількості брусків [9, табл. 75, с. 438].

Кількість брусків – 2; характеристики бруска – 23А, М28, СМ1; довжина бруска – 25 мм; поздовжнє суперфінішування; кут установлення – 60 °; ширина В = 8 мм.

Розрахунок режимів різання [9, с. 439].

Амплітуда коливань брусків – 2.6 мм, вибрано А = 5 мм.

Швидкість коливань $V_{кол}$ = 5…7 м/хв, взято 6 мм/хв. Коефіцієнт співвідношення колової швидкості обертання і швидкості коливального руху

$$K = \frac{V_{об}}{V_{кол}};$$

$$K_{чорн} = 3;$$

$$K_{чист} = 10;$$

$$V_{об.чорн} = 3 \cdot V_{кол} = 18 \text{ м/хв};$$

$$V_{об.чист} = 60 \text{ м/хв}.$$

Тиск для сталей – 0,1…0,3 МПа.

У нашому випадку – 0,2 МПа.

Основний час

$$T_o = \frac{C_T \cdot D_{обр}^{0.6}}{Ra^x} \cdot k \cdot \frac{l_{обр}}{l_{д.бр}}, \qquad (9.81)$$

де C_T = 0,067; x = 0,6; k = 0,62.

Тоді

$$T_o = \frac{0,67 \cdot 46^{0,6}}{0,16^{0,6}} \cdot 0,62 \cdot \frac{30}{25} = 1,5 \text{ хв}.$$

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Архангельский, А. Я. Учебник по классическим версиям Delphi . Программирование в Delphi / А. Я. Архангельский. – М. : ООО «Бином-пресс», 2006. – 1152 с.

2. Архангельский, А. Я. Delphi 7 : справ. пособие / А. Я. Архангельский. – М. : ООО «Бином-пресс», 2008. – 867 с.

3. Архангельский, А. Я. Delphi 2006. Язык Delphi. Классы функции WIN 32 и NET : справ. пособие / А. Я. Архангельский. – М. : ООО «Бином-пресс», 2006. – 732 с.

4. Архангельский, А. Я. Приемы программирования в Delphi на основе VCL. NET / А. Я. Архангельский. – М. : ООО «Бином-пресс», 2006. – 565 с.

5. Архангельский, А. Я. Язык паскаль и основы программирования в Delphi : учеб. пособие / А. Я. Архангельский. – М. : ООО «Бином-пресс», 2004. – 232 с.

6. Тайкера, Стив. Delphi 5. Руководство пользователя. Т. 1. Основные методы и технологии программирования : пер. с англ. / Стив Тайкера, Пачека Скавье. – М. : Изд. дом «Вильямс», 2001. – 832 с.

7. Тайкера, Стив. Delphi 5. Руководство пользователя. Т. 1. Основные методы и технологии программирования : пер. с англ. / Стив Тайкера, Пачека Скавье. – М. : Изд. дом «Вильямс», 2001. – 832 с.

8. Тайкера, Стив. Delphi 5. Руководство пользователя. Т. 2. Основные методы и технологии программирования: пер. с англ. / Стив Тайкера, Пачека Скавье. – М. : Изд. дом «Вильямс», 2001. – 992 с.

9. Система генерации баз знаний. – М. : Центр СПРУТ-Т, 2000. – 14 с.

10. Косилова, А. Г. Справочник технолога-машиностроителя. В 2 т. Т. 2 / А. Г. Косилова, Р. К . Мещерякова. – М.: Машиностроение, 1985. – 656 с.

11. Разработка маршрутных технологических процессов изготовле-ления авиадвигателей / В. Д. Сотников , А. И. Долматов, А. Ф. Горбачев, С. В. Яценко. – Харьков : ХАИ, 1989. – 41 с.

12. Барсуков, А. П. Исследование кинематической точности зубчатого колеса, обработанного на зубофрезерном станке : метод. указания по выполнению лаб. работ / А. П. Барсуков, А. Д. Некрасов, Б. С. Белоконь. – Харьков : ХАИ, 1984. – 52 с.

ЗМІСТ

Навчальне видання

Третяк Володимир Васильович

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕХНОЛОГІЧНИХ РОЗРАХУНКІВ В ОБ'ЄКТНО-ОРІЄНТОВАНОМУ СЕРЕДОВИЩІ

Редактор Н. М. Сікульська

Зв. план, 2021
Підписано до друку 30.11.2021
Формат 60х84 1/16. Папір офс. Офс. друк
Ум. друк. арк. 4,4. Обл.-вид.арк. 5. Наклад 100 пр.
Замовлення 277. Ціна вільна

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
"Харківський авіаційний інститут"
61070, Харків-70, вул. Чкалова, 17
http://www/khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
iszdat@khai.edu

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції сер. ДК № 391 від 30.03.2001