УДК 622.245

ПРО ОСОБЛИВОСТІ ЗАСТОСУВАННЯ МОВИ DELPHI ПІД ЧАС СТВОРЕННЯ ЕЛЕКТРОННОГО ДОВІДНИКА МЕТАЛОРІЗАЛЬНИХ ІНСТРУМЕНТІВ У НАФТОГАЗОВОМУ МАШИНОБУДУВАННІ

О.Р.Онисько, Л.О.Борущак, Ю.Д.Петрина, В.В.Врюкало

ІФНТУНГ, 76019, м. Івано-Франківськ, вул. Карпатська, 15, тел. (03422) 41166
e-mail: p u b l i c @ n u n g . e d u . u a

Одним із важливих завдань нафтогазового машинобудування є створення інструментальної бази для реалізації технологічних процесів обробки поверхонь деталей. До останніх відносяться бурові, аварійні інструменти та деталі обладнання промислів.

Суттєво знизити витрати на проектування і виробництво високоточних інструментів можна шляхом автоматизованого проектування. Основними етапами цього процесу є створення програм на мові Delphi з їх подальшим використанням для параметричного тривимірного моделювання необхідних інструментів у середовищі Autocad. Кінцевим результатом є робочі креслення останніх. Отримані проекти можна з високою продуктивністю реалізувати на сучасному обладнанні з числовим програмним керуванням. Перевагою такого проектування є висока точність геометричних параметрів інструментів, їх розмірів, а також гнучкість самої системи технологічної підготовки інструментального виробництва.

К числу важнейших задач нефтегазового машиностроения относится создание инструментальной базы для реализации технологических процессов обработки поверхностей деталей. К последним относятся буровые, аварийные инструменты и детали оборудования промыслов.

Существенно снизить затраты на проектирование и производство высокоточных инструментов можно путем автоматизированного проектирования. Основными стадиями этого процесса являются создание программ на языке Delphi с их дальнейшим использованием для параметрического трехмерного моделирования необходимых инструментов в среде Autocad. В конечном итоге получают рабочие чертежи инструментов. Полученные проекты можно с высокой производительностью реализовать на современном оборудовании с числовым программным управленим. Преимумуществом такого проектирования является высокая точность геометрических параметров инструментов, их размеров, а также гибкость самой системы технологической подготовки инструментального производства.

Creating of the cutting instrument basis for the improvement of the technological machining processes of machine details surfaces is one of the most important aim in the oil and gas machine building industry.

Automatic designing is the best path to product of the high precision instruments. To do one the 3-Dimention modeling of that instruments by applying of Delphi system programming together with AutoCAD is very useful. The digitally machine-tools working programs are opened for the projects received from that cooperation between Delphi – preparing application and AutoCAD 3D- designing. One of the preferences of this application is its using flexibility.

Актуальність проблеми та її стан. На сучасному етапі розвитку нафтогазового машинобудування та гірничовидобувної промисловості значне місце посідає проблема швидкого і ефективного виготовлення та ремонту чимраз ширшого спектру деталей для обладнання галузі. До цього переліку можна віднести інструменти – долота різних типів, в першу чергу, шарошкові, турбобури, аварійні інструменти та агрегати для пнемо - чи гідроударного буріння. Конструктивне виконання вказаної продукції передбачає наявність у неї складних фасонних поверхонь. До них відносяться поверхні різьбові (кріпильні) та напрямні, бігові доріжки доліт, поверхні, що контактують з робочим середовищем турбобурів тощо.

Технологічні процеси обробки вказаних поверхонь передбачають застосування, в першу чергу, різців. Останні можуть бути використані як різці для токарної обробки та як елементи фасонних (канавкових) фрез. Відповідно отримання високих показників якості обробки значною мірою залежатиме від точності різальних інструментів. Огляд інформаційних джерел в цьому напрямі свідчить, що конкретних досліджень з підвищення якості проектування металорізальних інструментів із використанням прикладних програм параметричного моделювання в Україні фактично не проводиться.

Постановка задачі дослідження. З метою підвищення точності та зниження витрат на проектування широкого спектру різцевих інструментів для нафтогазового машинобудування нами запропоновано комплексно використати можливості мови Delphi та конструкторського редактора Autocad**.** В результаті потрібно отримати прикладну програму проектування різців, параметричні тривимірні моделі необхідних інструментів та робочі креслення останніх. Вирішення поставленого завдання відобразимо на прикладі конструювання токарного різця.

Як відомо, обєктно-орієнтована мова Delphi грунтується на трьох основних принципах – інкапсуляції, поліморфізмі та спадковості [1]. Отож, об’єкти в середовищі Delphi можуть створюватися

користувачами – програмістами, як такі , що є видозміненими спадкоємцями раніше створених об'єктів. Оскільки всі об'єкти цієї мови описуються програмно, а характерні їм властивості є фрагментами таких програм, то, свого часу, розробниками Delphi було запропоновано поділ всіх об'єктів на класи. Відтак класи стали програмним вмістилищем конкретного набору спільних для певної групи об'єктів властивостей . Самі ж об'єкти стали екземплярами певного класу. Різниця між екземплярами одного і того ж класу полягає в наданні однаковим за назвою властивостям різних значень. Окрім властивостей в класах передбачено ще і методи їх застосування. Для прикладу, існує поняття геометричних параметрів різального клина – це кути: передній , задній і кут загострення , які функціонально пов'язані між собою спільною формулою

$$\alpha + \beta + \gamma = \pi / 2 .$$

Зрозуміло, що будь-яка зміна значення властивості β мусить призвести до зміни значень одного чи двох інших властивостей – α і γ. Це, в свою чергу, наводить на думку про недопустимість одночасної зміни значень усіх трьох параметрів. Отже, програмно повинно бути передбачена неможливість взаємної неузгодженості значень цих властивостей. Для забезпечення непорушності самої формули, тобто програмного захисту від недбалого втручання, яке призвело б до її зміни, варто скористатись вищезгаданим принципом інкапсуляції. Це означає, що програмно вказану формулу описано в класі і впливає на зміни значень властивостей, що в ній фігурують як доданки, але доступ до самої формули для програмістів-користувачів обмежений.

Оскільки всі лезові металорізальні інструменти обов'язково містять один чи багато різальних клинів, то варто скоритатись цим поняттям, створивши базовий для всіх цих інструментів клас – "клин".

Отже, створено клас Tklyn, що походить від базового класу мови Delphi– TObject. Для дотримання принципу інкапсуляції цей клас прописано в окремому модулі – Cut_unit. До членів класу входять геометричні параметри, як властивості, на мові Delphi вони називаються – property.

```
{не вказаний розділ private }
        public
property alfa:real read Falfa write Setalfa;
 {оголошення властивості – альфа (задній статичний кут в радіанах)}
property gama:real read Fgama write Setgama;
 {оголошення властивості – гама (передній статичний кут в радіанах)}
property beta:real read Fbeta write Setbeta;
{ оголошення властивості –бета (статичний кут загострення в радіанах)}
property lambda:real read Flambda write Setlambda;
 { оголошення властивості –лямбда (кут нахилу різальної кромки в радіанах)}
 property list:TStringList read Flist write Setlist;
{список геометричних параметрів}
function alfa_ins(alfa:real;lambda:real):real;
{функція визначення заднього інструментального кута}
function gama_ins(gama:real;lambda:real):real;
{функція визначення переднього інструментального кута}
function beta_ins(gama:real;alfa:real):real;
{функція визначення інструментального кута загострення}
 procedure Set_List;{процедура запису у список геометричних параметрів }
procedure SetAll(a,g,b,l:real);
{процедура встановлення всіх статичних геометричних параметрів різального клина}
procedure initialize;
{встановлення початкових геометричних параметрів, тобто їх ініціалізація "за замовчуванням"}
end;
```

В розділі implementation даного модуля описані вище задекларовані процедури і функції. Інкапсуляція формули

$$\alpha+\beta+\gamma =\pi/2$$

забезпечена в такий спосіб:

```
        procedure Tklyn.Setalfa(  const Value: real);
{інкапсульована процедура встановлення заднього статичного кута }
begin
  Falfa := Value;
  Fbeta:= (pi/2)-Falfa-Fgama;
  FGama:= (pi/2)-Fbeta-Fgama;
end;
```

Аналогічні процедури стосуються переднього кута і кута загострення.

Задекларовані функції визначення інструментальних геометричних параметрів створено доступними для майбутнього користувача-програміста і задекларовано в розділі публічних оголошень – public, а описано в розділі

```
        implementation:
function Tklyn.alfa_ins(alfa:real;lambda: real): real;
{функція визначення значення інструментального заднього кута}
begin
  result:=arctan((sin(alfa)/(cos(alfa))*cos(lambda)))*360/(2*pi);
end;
function Tklyn.gama_ins(gama:real;lambda: real ):real;
{функція визначення значення інструментального переднього кута}
begin
   result:=(arctan(sin(gama)/(cos(gama)*cos(lambda))))*360/(2*pi);
end;
function Tklyn.beta_ins(gama:real;alfa:real):real;
{функція визначення значення інструментального кута бета}
begin
  result:=90-arctan((sin(alfa)/(cos(alfa))*cos(lambda)))*360/(2*pi)-
(arctan(sin(gama)/(cos(gama)*cos(lambda))))*360/(2*pi);
end;
```

Оскільки клас TKlyn не є класом візуальних об’єктів, то для ефективного доступу до візуальних компонентів Delphi запропоновано властивість – List (список геометричних параметрів):

```
property list:TStringList read Flist write Setlist;
```

таі спеціальну процедуру запису, тобто долучення до списку відповідних параметрів:

```
        procedure Tklyn.Set_List;
var str1,str2,str3:string[49];
a:real;
begin
str(alfa_ins(alfa,lambda):4:2,str1);
str(gama_ins(gama,lambda):4:2,str2);
str(beta_ins(alfa,gama):4:2,str3);
list.Add('задній статичний кут+' '+FloatToStr(alfa*360/(2*pi))+' '+'град');
list.Add('передній статичний кут’ +' '+FloatToStr(gama*360/(2*pi))+' '+'град');
list.Add('статичний кут загострення'+' '+FloatToStr(beta*360/(2*pi))+' '+'град');
list.Add('кут нахилу різальної кромки'+' '+FloatToStr(lambda*360/(2*pi))+' '+'град');
list.Add('задній інструментальний кут'+' '+str1+' '+'град');
list.Add('передній інструментальний кут'+' '+str2+' '+'град');
list.Add(' інструментальний кут загострення'+' '+str3+' '+'град');
end;
```

Встановлення статичних геометричних параметрів може відбуватися шляхом виклику процедури SetAll:

```
procedure TKlyn.SetAll(a,g,b,l:real);
begin
   alfa:=2*pi*a/360; {в радіанах}
   gama:=2*pi*g/360;  { в радіанах }
   lambda:=2*pi*l/360; { в радіанах }
   beta:=2*pi*b/360;  { в радіанах }
end;
```

Надання значень фактичних параметрів вказаної процедури відбувається у градусах, що є більш зручним в її використанні фахівцями з програмування і різальних інструментів.

Оскільки клас Tklyn створено як спадкоємець класу TObject, то можна скористатись методом Create вказаного класу для створення екземпляру класу Tklyn. За цим принципом спадковості слід створити також екземпляр оголошеної властивості - property list:TStringList read Flist write Setlist.

Спадкоємцем класу Tklyn запропонованио ще один клас Tplankut. В цьому класі властивостями є кути в плані прямого прохідного різця, а також функції визначення повздовжніх переднього і задніх кутів і кут нахилу допоміжної різальної кромки. Оскільки різальна частина вказаного різця в тій чи іншій формі є інтегральною частиною багатьох інших багатолезових інструментів, то, очевидно, клас TplanKut відноситься до базових класів, за допомогою яких створюються параметричні моделі усіх лезових різальних інструментів.

```
TPlanKut=class(Tklyn)
{оголошення класу – Кути в плані }
public
property fi:real read Ffi write Setfi; {головний кут в плані}
property fi1:real read Ffi1 write Setfi1; {допоміжний кут у плані}
property epselon:real read Getepselon write Setepselon; {кут епсилон }
property alfa_d:real read Falfa_d write Setalfa_d; {допоміжний задній кут }
function alfa_p(ALFA,fi,lambda:real):real; {головний задній повздовжній кут }
function lambda1(gama,lambda,fi,fi1:real):real;{кут нахилу допоміжної різальної кромки }
function alfa_p1(alfa_d,fi1,lambda1:real):real; {допоміжний повздовжній задній кут }
function gama_p(gama,fi,lambda:real):real;{повздовжній передній кут}
procedure SetAll (a,g,l,f,f1,a_d:real);overload;
procedure Set_List; overload;
end;
```

Процедури SetAll і Set_List як видно по директиві overload є спадкоємцями аналогічних процедур класу Tklyn. Принцип поліморфізму мови Дельфі сприяє ідентифікації об'єктами, що створені на базі класів Tklyn чи TplanKut властивих саме їм одноіменних за назвою процедур.

Варто розглянути ще раз розділ implementation модуля Cut_unit, в тій його частині, що містить опис функцій – членів класу TplanKut. Згідно з принципом інкапсуляції, по аналогії з геометричними параметрами різального клина створено і процедура надання значення куту іпсилон:

```
procedure TPlanKut.SetEpselon( Value: real);
begin
  Fepselon := Value;
  Ffi:=2*Pi-Ffi1-Fepselon;
  ffi1:=2*Pi-Ffi-Fepselon;
end;
```

Тож зміна значення цього кута автоматично призведе до зміни головного і допоміжного кутів у плані.

Як вже згадувалося процедура SetAll є спадкоємницею одноіменної процедури класу–попередника і містить ряд додаткових параметрів:

```
procedure TPlanKut.SetAll (a,g,l,f,f1,a_d:real);
begin
  inherited SETALL(a,g,l);
  fi:=2*pi*f/360;
  fi1:=2*pi*f1/360;
  alfa_d:=2*pi*a_d/360;
end;
```

Останнім класом, який міститься в модулі cut_unit є клас, що уможливлює створення параметричної моделі прямого прохідного різця. Цей клас названо Trizets. Ось його декларація:

```
TRizets=class(TPlanKut) {не вказаний розділ privat}
Public
Property Program_Text:TStringList read FProgram_Text write SetProgram_Text;
Property length:real read Flength write Setlength; {довжина державки різця}
Property width:real read Fwidth write Setwidth; {ширина державки }
Property height:real read Fheight write Setheight; {висота державки різця}
Procedure SET_derjavca(l,w,h:real); { процедура встановлення розмірів державки різця}
```

```
Function vershyna_A(y:real):real; { визначення координати У вершини різця А}
Procedure Program_write(y,h:real);  { процедура запису лісп-програми }
Procedure Program_send; {процедура запису лліcп-програми у текстовий файл}
end;
```

Для створення Лісп-програми, тобто для параметричного проектування різця в середовищі Автокад запропоновано відповідну процедуру:

```
Procedure TRizets.Program_write(y,h:real);
{h – відстань в мм вершини різця від верха державки по вертикалі}
var sLength: string[5]; sWidth, sHeight: string[5]; {рядкові значення довжини, ширини і висоти державки }
 sFi, sAlfa_p, sGama:string[4]; {рядкові значення головного кута в плані, головного повздовжнього заднього кута, переднього кута}
 sVershyna_A: string[3]; {рядкове значення координати у вершини різця А}
 alfap:real; {визначення головного повздовжнього заднього кута в градусах}
 z,x,y1:real;{координати z,x,y використані в побудові передньої поверхні}
 sX,sZ,sY:string[3];{рядкові значення координат x, z}
 sLambda:string[5];{рядкове значення кута нахилу різальної кромки у градусах}
 b:real; sB:string[4];
 begin
 Program_text.Clear;
  alfap:=Alfa_p(ALFA,fi,lambda)*360/(2*pi);{визначення головного повздовжнього заднього кута у градусах }
str(length:5:1,sLength); {рядкове значення довжини державки}
str(width:5:1,sWidth); {рядкове значення ширини державки}
str(height:5:1,sHeight); {рядкове значення висоти державки}
str((Fi*360/(2*pi)):5:1,sFI); {рядкове значення головного кута в плані
str(Alfap:4:1,sAlfa_p); {рядкове значення головного повздовжнього заднього кута}
str((vershyna_A(y)):3:1,sVershyna_A);{рядкове значення координати у вершини А різця}
With Program_text do
 begin
 Add('(defun C:rizets()');
  {лісп –програма побудови прямого прохідного різця}
 add('(command "box" "0,0,0" "l" '+'"'+ sLength +'" "' +sWidth+ '" "'+sHeight+'") ');
  {лісп-програма побудови головної різальної кромки }
 add('(command "box" "0,' +  sVershyna_A +',' +sHeight +'" "-40,-20,-30") ');
 add('(command "ROTATE3D" "last" "SI" "y" "0,'+sVershyna_A+','+sHeight+'" "-'+sAlfa_p+'")');
 add('(command "subtract" "'+ sLength+','+ sWidth+','+ sHeight+'" "SI" "last" "SI")');
 add('(command "box"  "0,' +  sVershyna_A +',' +sHeight +'"  "-40,-20,-30" )');
 add('(command "rotate" "last" "SI" "0,'+sVershyna_A+','+sHeight+'" "'+sFi+'")');
 add('(command "ROTATE3D" "last" "SI" "y" "0,'+sVershyna_A+','+sHeight+'" "-'+sAlfa_p+'")');
 add('(command "subtract" "'+ sLength+','+ sWidth+','+ sHeight+'" "SI" "last" "SI")');
 end;
   y1:=Vershyna_A(y)+(width-Vershyna_A(y))*1.2/cos(Fi1);
   str(y1:4:1,sy);
With Program_text do
 begin
  { лісп-програма побудови  допоміжної різальної кромки }
 add('(command "box" "0,'+sVershyna_A+','+sHeight+'" "-40,'+sy+',-30")');
 add('(command "ROTATE3D" "Last" "SI" "y" "0,'+sVershyna_A+','+sHeight+'" "-'+sAlfa_p+'")');
 add('(command "subtract" "'+ sLength+','+ sWidth+','+ sHeight+'" "SI" "last" "SI")');
 add('(command "box" "0,'+sVershyna_A+','+sHeight+'" "-40,'+sy+',-30")');
 add('(command "rotate" "last" "SI" "0,'+sVershyna_A+','+sHeight+'" "-'+sFi+'")');
 add('(command "ROTATE3D" "last" "SI" "y" "0,'+sVershyna_A+','+sHeight+'" "-'+sAlfa_p+'")');
 add('(command "subtract" "'+ sLength+','+ sWidth+','+ sHeight+'" "SI" "last" "SI")');
  end;
{визначення кута нахилу головної різальної кромки}
 b:=(vershyna_A(y))*(sin(fi));STR(b:4:1,sB);{ визначення координати z першої точки “box” для повернутих координат }
      str(width*(cos(fi)):4:1,sWidth);{ визначення координати x першої точки “box” для повернутих координат }
      z:=1.2*((width-vershyna_A(y))*(cos((pi/2)-epselon))/(cos(Fi1))+b);{ визначення координати z другої точки “box” для повернутих координат }
```

```
        x:= (vershyna_A(y))*((1/(cos(Fi)))-cos(fi))+z*(cos((pi/2)-fi))/(cos(fi));{ визначення координати x другої точки “box” для повернутих координат }
        y1:=height+20; {визначення координати у другої точки “” для повернутих координат}
        str(x:4:1,sX); str(z:4:1,sZ); str(y1:4:1,sy);
        lambda:=lambda*360/(2*(pi));
        str(lambda:4:1,sLambda); {рядкове значення кута нахилу різальної кромки в градусах}
      with Program_text do
      begin
     {лісп-програма побудови передньої програми з урахування значення кута нахилу різальної кромки}
      add('(command "ucs" "z" "-'+SFi+'")');{повернута вздовж головної різальної кромки вісь x}
      add('(command "ucs" "x" "90")');
        {визначення координат “box” в повернутих щодо головної різальної кромки осях}
      add('(command "box" "-'+sWidth+','+sheight+',0" "'+sx+','+sy+',-'+sz+'")');
      end;
        {визначення координат вершини різця A в повернутій вздовж проекції різальної кромки на основну площину системі координат до і після її переміщення на висоту h щодо верха державки різця}
        y1:=height-h;str(y1:4:1,sY);{значення координати у після зниження вершини A на розмір h}
        x:=(vershyna_A(y))*(cos(fi));Str(x:4:1,sX);{координата x вершини різця до переміщення}
        z:=h*cos(alfa_p(ALFA,fi,lambda));
        z:=z*cos((pi/2-fi));
        z:=x-z;str(z:4:1,sZ);{координата x вершини різця A після переміщення на розмір h}
        with Program_text do
      begin
        {лісп_програма встановлення кута нахилу різальної кромки}
      add('(command "move" "last" "Si" "'+sX+','+sHeight+',0" "'+sX+','+sY+',+'0")');
      add('(command "rotate" "last" "si" "'+sz+','+sy+',0" "'+'-'+sLambda+'")');
      end;
        {встановлення координат переміщеної вершини різця у світовій системі координат }
       z:=height-h;str(z:4:1,sZ);{координата z вершини різця після її переміщення на розмір h }
       x:=h*sin(alfa_p(ALFA,fi,lambda))/cos(alfa_p(ALFA,fi,lambda));str(x:4:1,sX);
       gama:=(gama*360)/(2*pi);str(gama:4:1,sGama);
       {лісп_програма встановлення переднього кута}
      with Program_text do
      begin
      add('(command "ucs" "w")'); {повернення в світову систему координат}
      add('(command  "ROTATE3D" "Last" "SI" "y" "'+sX+','+sWidth+','+sZ+'" "'+sGama+'")');
      end;
        str(width:4:1,sWidth);
       with Program_text do
      begin
      add('(command "Subtract" "'+sLength+','+sWidth+','+sHeight+'" "SI" "Last" "SI")');
      add('(princ))');
      end;
    end;
```

Отож, подана процедура уможливила надання властивості Program_Text відповідного до заданих користувачем парметрів вигляду , а саме укладених в неї у відповідній послідовності рядків лісп-програми з конкретними даними.

Серед задекларованих у класі Trizets методів міститься процедура запису лісп-програми у текстовий файл. В результаті її застосування отримується текстовий файл з розширенням *lsp, який і є програмою побудови різця на мові АвтоЛісп. Ось її форматний вивід за таких вхідних параметрів:

```
розміри державки довжина - 100 , ширина – 30, висота - 40
Значення величини заднього кута – 12 0
значення величини переднього кута – 8 0
значення величини кута нахилу різальної кромки 7 0
зниження вершини різця A відносно верха державки - 9
(defun C:rizets()
(command "box" "0,0,0" "l" "100.0" " 30.0" " 40.0")
(command "box" "0,18., 40.0" "-40,-20,-30")
(command "ROTATE3D" "last" "SI" "y" "0,18., 40.0" "-17.2")
```

```
(command "subtract" "100.0, 30.0, 40.0" "SI" "last" "SI")
(command "box"  "0,18., 40.0"  "-40,-20,-30" )
(command "rotate" "last" "SI" "0,18., 40.0" " 45.")
(command "ROTATE3D" "last" "SI" "y" "0,18., 40.0" "-17.2")
(command "subtract" "100.0, 30.0, 40.0" "SI" "last" "SI")
(command "box" "0,18., 40.0" "-40,38.,-30")
(command "ROTATE3D" "Last" "SI" "y" "0,18., 40.0" "-17.2")
(command "subtract" "100.0, 30.0, 40.0" "SI" "last" "SI")
(command "box" "0,18., 40.0" "-40,38.,-30")
(command "rotate" "last" "SI" "0,18., 40.0" "- 45.")
(command "ROTATE3D" "last" "SI" "y" "0,18., 40.0" "-17.2")
(command "subtract" "100.0, 30.0, 40.0" "SI" "last" "SI")
(command "ucs" "z" "- 45.")
(command "ucs" "x" "90")
(command "box" "-21.2, 40.0,0" "48.,60.,-35.")
(command "move" "last" "Si" "12., 40.0,0" "12.,31.,0")
(command "rotate" "last" "si" " 6.,31.,0" "- 7.0")
(command "ucs" "w")
(command  "ROTATE3D" "Last" "SI" "y" " 3.,21.2,31." " 8.0")
(command "Subtract" "100.0,30.0, 40.0" "SI" "Last" "SI")
(princ))
```

Керуюча прикладна програма створена на основі класів Tklyn, Tplankut, TRizets може мати доволі лаконічний вигляд, наприклад такий, що представлено на рисунку1.

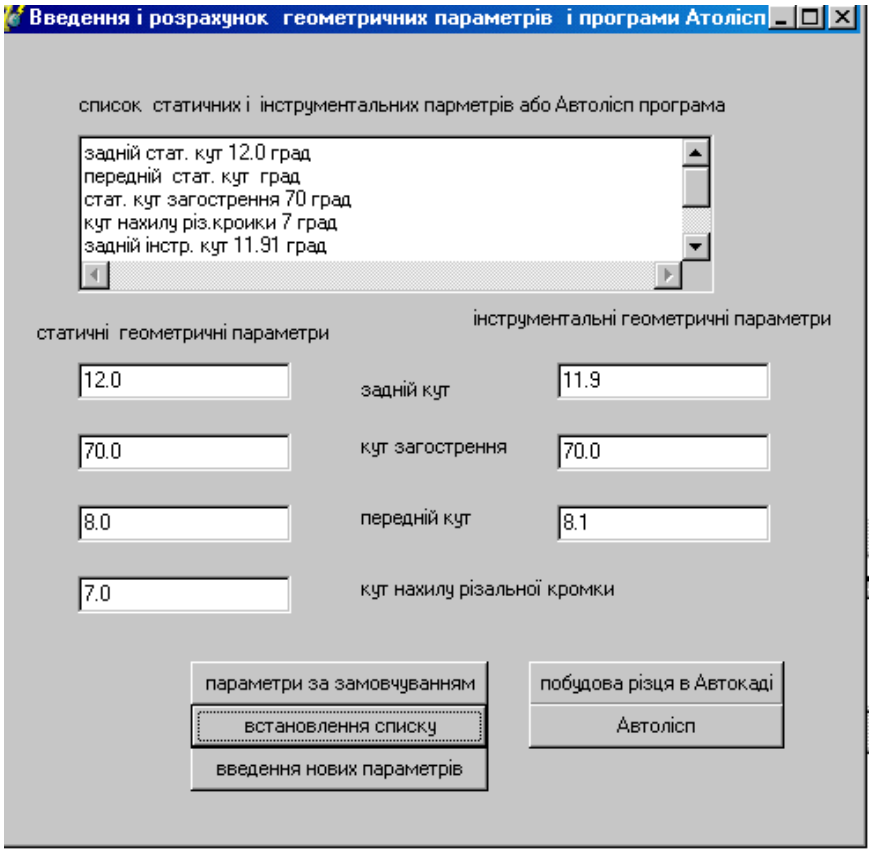

Рисунок 1 — Зовнішній вигляд прикладної програми розрахунку і побудови прохідного прямого різця

Висновок. Кінцево вартісним є не просто отримання числових даних інструментальних геометричних параметрів, вкрай потрібних для вимірювання і контролю різального клина, а власне формування програми Автоліспа для побудови твердотільної моделі різця із максимально можливою для галузі точністю отримання розмірів. Отримана модель і програма для неї уможливлює якнайточніше і найпродуктивніше відтворення моделі на металорізальному обладнанні з цифровим виконанням в натурі. На рисунку 2 зображено твердотільну модель розрахованого за введеними параметрами в прикладній програмі (рис. 1).

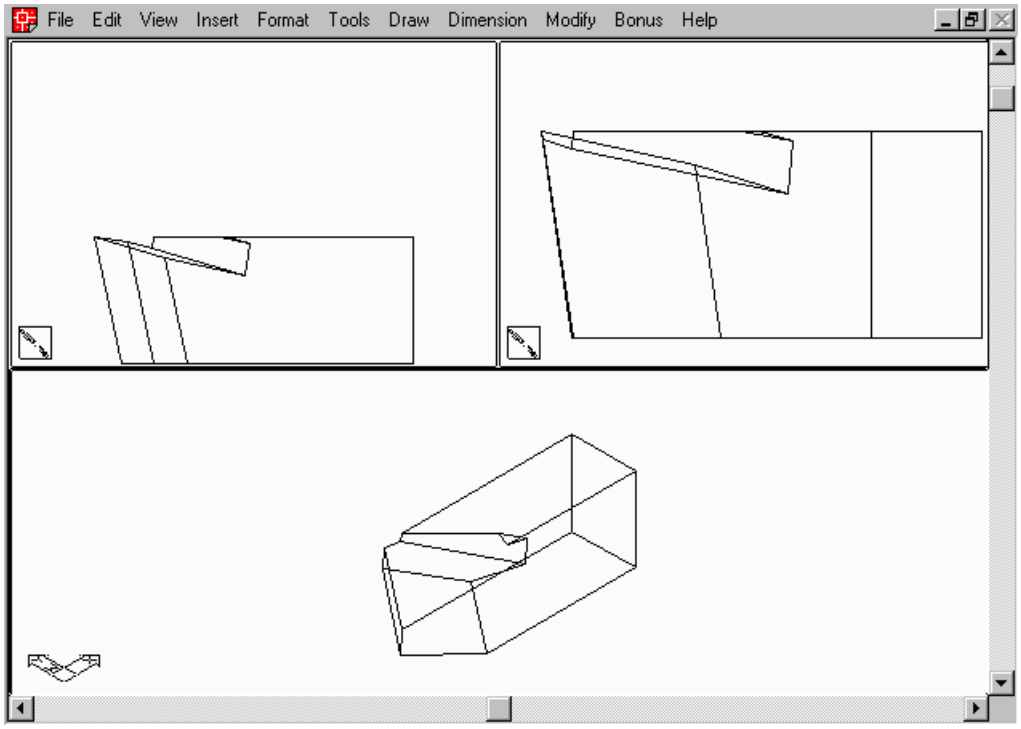

Рисунок 2 — Тривимірна твердотільна модель різця з параметрами, що введені і розраховані у показаній на рисунку 1 прикладній програмі

Література

1 Глинський Я.М.Паскаль. TurboPascal I Delphi. [текст] / Я.М.Глинський, В.Є.Анохін, В.А.Ряжська. – Львів: Деол, 2002. – 144 с.

Стаття поступила в редакційну колегію
18.05.09
Рекомендована до друку професором
Мойсишиним В.М.