Ćwiczenie nr 14 – Zaawansowane możliwości programu

Program AutoCAD jest aplikacją wspomagającą projektowanie. Wyposażony jest on w polecenia umożliwiające korzystanie z wyników obliczeń innych programów oraz z własnego kalkulatora.

Obliczenia – wykorzystanie kalkulatora

Program AutoCAD jest aplikacją wspomagającą projektowanie. Wyposażony jest on w polecenie **kalk**, które przywołuje kalkulator geometryczny. Można je przywołać w trakcie działania innego polecenia stawiając przed nazwą znak apostrofu czyli tak **'kalk.** Po jego wywołaniu ukazuje się napis:

```
>> Wyrażenie:
```

po którym wpisujemy wyrażenie na zasadach podobnych jak w Pascalu albo lepiej w Basicu. np. obliczenie pola okręgu o promieniu 2,5 (czyli $2{,}5^2\pi$) zapiszemy tak:

```
>> Wyrażenie: 2.5^2*pi
```

Możemy tu korzystać z operatorów (wymienione w kolejności rosnącego priorytetu):

Dodawanie, odejmowanie:	+ –
Mnożenie, dzielenie:	* /
Potęgowanie:	^ (np. 2.5^0.5 = $2{,}5^{0.5}$)

Nawiasy okrągłe ( ) służą do zmiany kolejności wykonywania obliczeń. W wyrażeniach można użyć następujących funkcji:

Funkcja	**Nazwa**
Trygonometryczne	`sin(a)`, `cos(a)`, `tang(a)`
Trygonometryczne "arcus"	`asin(x)`, `acos(x)`, `atan(x)`
Logarytm naturalny i dziesiętny:	`ln(x)`, `log(x)`
Potęga e i potęga 10:	`exp(x)`, `exp10(x)`
Kwadrat i pierwiastek liczby:	`sqr(x)`, `sqrt(x)`
Zamiana radianów na stopnie i odwrotnie	`r2d(a)`, `d2r(a)`
Liczba π	`pi` (*symbol specjalny predefiniowany*)
Pobranie promienia okręgu łuku	`rad` (*prosi o wskazanie okręgu lub łuku*)
Zaokrąglenie do najbliższej liczby całkowitej	`round(x)`

Tutaj *x* – liczba lub wyrażenie rzeczywiste; *a* – liczba lub wyrażenie określające kąt w stopniach dziesiętnych. Liczby podajemy jak w Pascalu np. 10 -20.45 10.34E5 itp. Jeżeli kąty chcemy podać w innych jednostkach to robimy to tak np. : w radianach – 6.23**r**; w gradach – 120.56**g**; w stopniach, minutach i sekundach – 12**d**30**'**45**"**.

Kalkulator wykonuje też obliczenia na punktach i wektorach, które zapisywane są identycznie. Oto sposoby ich zapisywania:

Układ	**Format**	**Przykład**
prostokątny	`[x,y,z]` lub `[x,y]`	`[2,1,0] [1+1,1,0]`
biegunowy	`[r<α]`	`[100.0<45] [20*5<arcsin(0.5)]`
walcowy	`[r<α,z]`	`[50.23<33d'50,-46]`
sferyczny	`[r<α<ϕ]`	`[4.5<0.6r<33]`

Użyte symbole są **wyrażeniami** reprezentującymi: *x*, *y*, *z* – współrzędne; *r* – promień i α, ϕ – kąty. Dla punktów i wektorów przewidziano też specjalne funkcje i operatory:

Operacja	Zapis/przykład
Dodawanie odejmowanie wektorów	`+ -` (np. `[1,0,0]+[2,0,1]`)
Mnożenie skalarne wektorów lub wektora przez liczbę	`*` (*np.* `2*v` *lub* `v * u`)
Dzielenie wektora przez liczbę	`/` (*np.* `v/2.5`)
Wektorowe mnożenie wektorów	`&` (*np.* `v&u  [1,0,0]&[0,1,0]`)
Obliczanie długości wektora lub wart bezwzględnej	`abs(v)`
Wektor i wektor jednostkowy między punktami	`vec(A,B)` `vec1(A,B)`
Odległość między punktami	`dist(A,B)`
Wyznacza punkt na linii **AB**. Parametr t definiuje pozycję punktu na linii. x=0 oznacza punkt p1, x=1 oznacza punkt p2 a np. x=0.5 oznacz środek okręgu	`plt(A,B,x)`

Kąt miedzy **v** a osią OX
Kąt miedzy odcinkiem **AB** a osią OX
Kąt o wierzchołku **A** między **AB** i **AC** czyli ∠**ABC**
	`ang(v)`
	`ang(A,B)`
	`ang(A,B,C)`

Tutaj **v, u** oznaczają wektory a **A**, **B** lub **C** oznaczają punkty zapisane symbolicznie lub w formacie [..,..,..]. Dodatkowo do wprowadzania punktów możemy użyć symbolu @ – dla ostatnio wprowadzonego punktu oraz funkcji **cur** – kiedy punkt trzeba wskazać kursorem. Aby pobrać odpowiednie punkty charakterystyczne obiektów użyj funkcji **end** – dla punktu końcowego; **ins** – dla punktu wstawienia bloku; **cen, mid** – dla punktu środkowego i symetrii; **qua** – dla punktu kwadrantowego; **per, tan** – dla punktu prostopadłego i stycznego; **int** – dla punkt przecięcia oraz **nea** – dla punktu bliskiego. We wszystkich powyższych przypadkach użycie symbolu w wyrażeniu powoduje zatrzymanie obliczeń i poproszenie użytkownika o wskazanie albo punktu dla funkcji **cur**, albo obiektu w pozostałych przypadkach.

W wyrażeniach możemy zdefiniować też własne symbole i przypisać im jakąś wartość. Wystarczy przed wyrażeniem napisać nazwę zmiennej i znak „=". Nazwy tej później można użyć w odpowiedzi na żądania AuotCAD'a. W takim przypadku symbol poprzedzamy znakiem „!". Dokładny opis działania kalkulatora znajdziesz systemie pomocy (**F1 – Opis poleceń – Polecenia K – KALK**).

Przykłady

Wywołując polecenie nakładkowo mamy możliwość automatycznego wykorzystania uzyskanego wyniku jako odpowiedzi na pytanie AutoCAD'a. Na przykład, aby narysować koło o obwodzie 100 jednostek wykonujemy następująca sekwencję poleceń z klawiatury

```
Polecenie: okrąg
Określ środek okręgu lub [3p/2p/Ssr (sty sty promień)]: wskazujemy punkt na ekranie
Określ promień okręgu lub [śreDnica]: d
Określ średnicę okręgu: 'cal (nakładkowe wywołanie kalkulatora)
>> Wyrażenie: 100/pi (obliczamy średnicę)
31.831 (ten wynik jest użyty jako odpowiedź na pytanie o srednicę)
```

Tu zapis **100/pi** oznacza wyliczenie średnicy na podstawie obwodu wg wzoru $D=B/\pi$ (B – obwód.). Zadanie to można wykonać też za pośrednictwem własnego symbolu:

```
Polecenie: cal
>> Wyrażenie: s = 100/pi (definiujemy symbol s i przypisujemy mu wynik wyrażenia)
31.831
Polecenie: okrąg
Określ środek okręgu lub [3p/2p/Ssr (sty sty promień)]:
Określ promień okręgu lub [śreDnica]: d
Określ średnicę okręgu: !s (używany symbol s jako odpowiedź na pytanie o wart. średnicy)
31.831
```

Następny przykład pokazuje jak złapać punkt (w czasie rysowania odcinka) leżący w 1/4 odległości między wskazanymi punktami P1 a P2. (**Uwaga** – wyłącz stałe tryby lokalizacji OBIEKT i korzystaj tylko z chwilowych)

```
Polecenie: _line Określ pierwszy punkt: 'cal
>> Wyrażenie: plt(cur,cur,0.25)
>> Podaj punkt:  (wskazujemy jakiś punkt P1 skutek 1-szego wywołania funkcji cur)
>> Podaj punkt:  (wskazujemy jakiś punkt P2 skutek 2-giego wywołania funkcji cur)
(475.585 440.662 0.0)
```

Następny przykład pozwala obliczyć kąt między trzema punktami wskazywanymi w kolejności wierzchołek, koniec pierwszego ramienia, koniec drugiego ramienia. Zmierzony kąt jest potem przypisywany do zmiennej *a* i można go wykorzystać przy rysowaniu np. łuku. (Patrz uwaga jak wyżej)

```
Polecenie: cal
>> Wyrażenie: a = ang(cur,cur,cur)
>> Podaj punkt:  (wskazujemy wierzchołek kąta - skutek 1-szego wywołania funkcji cur)
>> Podaj punkt:  (wskazujemy koniec 1-szego ramienia - skutek 2-giego wywołania funkcji cur)
>> Podaj punkt:  (wskazujemy koniec 2-giego ramienia - skutek 3-ciego wywołania funkcji cur)
63.5852
```

Inny przykład. Narysować przekrój kanału kołowego, przez który ma płynąć medium z prędkością 0,5 m/s z wydatkiem 20 m^3/h.

```
Polecenie: cal
>> Wyrażenie: v = 0.5  (nadajemy zmiennej v wartość prędkości medium)
0.5
Polecenie: cal
>> Wyrażenie: q = 20/3600  (nadajemy zmiennej Q  wartość przeliczoną na m/s)
0.00555556
Polecenie: cal
>> Wyrażenie: a = q/v  (obliczmy pole przekroju i wstawiamy do a)
0.0111111
Polecenie: cal  (obliczymy promień w [mm]  z zaokrągleniem do liczby całkowitej i wstawimy do R)
>> Wyrażenie: r = round( sqrt(a/pi)*1000 )
59
Polecenie: okrąg  (rysujemy okrąg)
Określ środek okręgu lub [3p/2p/Ssr (sty sty promień)]:  (wskazujemy jakiś punkt)
Określ promień okręgu lub [śreDnica] <50.3740>: !r  (korzystamy z r)
59
```

AutoLISP – podstawy

AutoLISP jest rozbudowaną wersją języka LISP, który jest językiem do przetwarzania list (ang. LISt Processing). Podstawowymi elementami są *lista* oraz *atom*. Są następujące rodzaje atomów:

1. Liczby całkowite np. 100 -456 67 itd.
2. Liczby rzeczywiste np. –12.45 10.34E-6 1E5 itd.
3. Łańcuchy tekstowe (napisy ujęte w cudzysłów) np. ”Podaj punkt:”
4. Symbole np.: nil T sqr a promien 1+ / * itd.

W łańcuchach tekstowych znak „\” pełni specjalną funkcję. Przy jego pomocy wprowadza się do napisu specjalne symbole np.: \n – znak nowej linii; \t – znak tabulacji; \” – znak cudzysłowu czy wreszcie \\ – sam znak ukośnika. O tym ostatnim należy pamiętać podając ścieżki do plików. Powinno to wyglądać np. tak: **”d:\\student\\acad\\test.lsp”.** Symbole pełnią rolę nazw zmiennych lub funkcji i w przeciwieństwie do większości języków mogą zaczynać się od cyfry jak np. standardowy symbol inkrementacji **1+** lub być znakami nie alfanumerycznymi jak np. ***** czy **/=** i inne.

Lista jest zbiorem elementów list i/lub atomów ujętych w nawiasy okrągłe i oddzielonych spacjami o ile sąsiadujące elementy nie są listami. Oto przykłady:

Lista pusta: **( )**
Listy 1-elementowe: **(a); (2.45); ((a b)) ("Wskaż obiekt:") (getstr)**
Listy 2-elemntowe: **(1.23 –67.4); (a (b c)); ((2 3 c) (b g (1 y)))**
Listy 3-elementowe: **(12.3 –56.6 78); (a (b c) d)** itd...

Zasada jest następująca. Jeżeli AutoCAD napotka listę to traktuje ją jak wyrażenie przy czym pierwszy element listy jest traktowany jako nazwa funkcji a pozostałe jako wyrażenia oznaczające parametry. Wywołanie funkcji w AutoLISPie wygląda więc tak:

```
(nazwa  par1  par2  par3 ...)
```

W przeciwieństwie do np. Pascala gdzie zapisano by to tak:

```
nazwa(par1, par2, par3...)
```

W Lispie nie ma deklaracji, instrukcji, definicji itp. Tu są tylko listy, które są wyrażeniami. Elementy takie jak definicja funkcji czy instrukcja są realizowane jako listy czyli wyrażenia. **Punkty AutoCAD'a są listami trzyelementowymi** w postaci (*x y z*). Poniżej podano kilka funkcji, które mogą być pomocne przy tworzeniu prostych makr.

Funkcja	Opis
`(+ a b c ...)`	Dodaje wyrażenia a, b, c ... równoważne a+b+c+...
`(- a b c ...)`	Odejmuje wyrażenia a, b, c ... równoważne a–b–c–... = a – (b+c+...)
`(* a b c ...)`	Mnoży wyrażenia a, b, c ... równoważne a*b*c*...
`(/ a b c ...)`	Dzieli wyrażenia a, b, c ... równoważne a/b/c/... = a/(b*c*...)
`(sin a)`	Sinus = sin(a)
`(cos a)`	cosinus = cos(a)
`(atan a)` `(atan a b)`	Arcustangens = arctg(a) lub = arctg(a/b)
`(sqrt a)`	pierwiastek kwadratowy z a
`(setq s1 w1 s2 w2 ...)`	Przypisanie symbolowi s1 wartości w1 symbolowi s2 wartości w2 itd..
`(getreal "text")`	Pobranie liczby z klawiatury łańcuch "text" jest opcjonalnym napisem wyświetlanym przed pobraniem.
`(getdist p "text")`	Pobiera (mierzy) odległość miedzy dwoma punktami. Oba parametry opcjonalne. Jeżeli punkt p nie jest podany to należy wskazać dwa punkty.
`(getpoint p "text")`	Pobiera punkt. Oba parametry opcjonalne. Jeżeli punkt p jest podany to na ekranie jest wleczony odcinek.
`(getcorner p "text")`	Pobiera punkt. Drugi parametr opcjonalny. Na ekranie jest wleczony jest prostokąt między punktem p a punktem wskazywanym przez kursor.
`(getangle p "text")`	Pobiera (mierzy) kąt między dwoma punktami a osią OX. Oba parametry opcjonalne. Jeżeli punkt p nie jest podany to należy wskazać dwa punkty.
`(progn a b ...)`	Łączy wiele wyrażeń w jedno podobnie jak begin .. end w Pascalu.
`(princ "text")`	Wypisuje napis "text"
`(ssget)`	Pobiera zbiór wyboru – wyświetla się napis Wybierz obiekty:

`(car p)`
`(cadr p)`
`(caddr p)`
	Wyciąga odpowiednio pierwszą,
	drugą
	i trzecią współrzędną z punktu p

`(list a b c ..)`	Tworzy listę z elementów a, b, c … czyli (a b c ...)
`(quote a)` lub `'a`	Zabrania obliczania wyrażenia a tzn. ma ono być traktowane dosłownie.
`(defun zrobcos (..)` `...)`	Definiuje nową funkcję o nazwie zrobcos.
`(princ a)`	Wypisuje na ekranie wartość związaną z a
`(terpri)`	Przenosi kursor testowy do następnego wiersza.
`(command ...)`	Wywołuje polecenie AutoCAD'a

Ten zestaw funkcji jest wystarczający, aby zbudować proste makra. Dokładny opis znajduje się (po angielsku) w systemie pomocy. Zainteresowanych odsyłam do literatury polskojęzycznej dostępnej w księgarniach informatycznych. Oto kilka przykładów:

(setq r 14.56) – przypisuje zmiennej *r* wartość 14,56
(setq p (list 10 0 0)) – przypisuje *p* listę (10 0 0). *P* jest teraz punktem (10,0,0)
(setq p '(l10 0 0)) – robi to samo, co wyrażenie wyżej.
(setq a (* (+ 1 2)(- 3 4))) – oblicza wyrażenie *(1+2)(3-4)* i wstawia je do zmiennej *a*.
(setq p (getpoint "Wskaż punkt")) – prosi użytkownika o wskazanie punktu i przypisuje go zmiennej *p*.
(command "linia" '(0 100) '(100 100) "") – rysuje poziomą linię "linia" między punktami (0, 100) a (100, 100) i kończy polecenie "".
(/ (* angle 180.0) pi) – przeliczenie radianów na stopnie dziesiętne wg wzoru $180\alpha/\pi$. Przeliczana wartość znajduje się w zmiennej *angle*.

Poniżej przykład wyrażenia, które podaje w stopniach nachylenie hipotetycznej prostej poprowadzonej między dwoma wskazanymi punktami a osią OX (wykorzystano tu ostatnie wyrażenie z powyższych przykładów). Dla przejrzystości zapisano ją w wielu wierszach i opatrzono komentarzem w Lispie komentarz zaczyna się od znaku średnika „;"

```
(progn
   ; Wyświetlamy napis informujący
   (princ "Pomiar kąta")
   ; Kursor do następnego wiersza
   (terpri)
   ; pobieramy kąt i przeliczamy go na stopnie
   (setq alfa (/ (* (getangle "Wskaż 1-szy punkt:") 180.0) pi))
   (terpri)
   ; Organizujemy wypisanie wartości kąta
   (princ "Zmierzony kąt = ")
   (princ alfa)
   (princ " stopni")
   ;"sztuczka" która powoduje ciche zakończenie polecenia
   (princ)
)
```

Dalej to samo wyrażenie, ale przekształcone w funkcję AuotLISP'a:

```
(defun kąt ()
  (setq alfa (/ (* (getangle "Wskaż 1-szy punkt:") 180.0) pi))
  ; ostatnie wyrażenie ustala zawsze wynik funkcji
  alfa
)
```

Oto przykład funkcji, która rysuje prostokąt.

```
(defun c:prost (/ p1 p2 p3 p4 x1 x2 y1 y2)
 ;Pobieramy narożniki prostokąta
   (setq p1 (getpoint "Wskaż 1-szy narożnik:"))
   (terpri)
   (setq p2 (getcorner p1 "Wskaż 2-szy narożnik:"))
 ;Odczytujemy współrzędne okr. granice prostokąta
   (setq
       x1 (car p1)
       x2 (car p2)
       y1 (cadr p1)
       y2 (cadr p2)
   )
```

```
  ;Tworzymy brakujące narożniki prostokąta
    (setq p3 (list x2 y1)  p4 (list x1 y2))
  ;Poleceniem LINE rysujemy prostokąt
    (command "_line" p1 p3 p2 p4 "_C")
    (princ)
)
```

W AutoLISPie obowiązuje zasada wg której nazwy funkcji Lispa zaczynające się od przedrostka **C:** a więc posiadające ogólną postać **C:XXXX** są traktowane jak polecenia AutoCAD'a o nazwie **XXXX**. Zatem powyższa definicja funkcji definiuje nowe polecenie o nazwie **PROST**. W nawiasach tuż po nazwie funkcji podaje się wykaz parametrów formalnych i zmiennych lokalnych. Wykazy te są oddzielone ukośnikiem „/". W powyższym przykładzie mamy tylko wykaz zmiennych lokalnych i są nimi *p1 p2 p3 p4 x1 x2 y1 y2*. Jeśli użyjemy symbolu i nie umieścimy go na tej liście to stanie się on symbolem globalnym, który można użyć po zakończeniu funkcji. Taka sytuacja ma miejsce w przypadku funkcji (kąt). Po jej wywołaniu zmierzony kąt jest zapamiętany w zmiennej *alfa*. Zmienną tę można później przywołać (pisząc **!alfa**) w odpowiedzi na pytania AutoCAD'a

Współpraca AutoLISP i AutoCAD.

Zasady współpracy są następujące. Wyrażenia (tylko te, które zaczynają się od nawiasu) AutoLISP'a można wpisywać bezpośrednio w linii poleceń a także w odpowiedzi na zapytania AutoCAD'a. Zmienne (symbole) wypisujemy poprzedzając je znakiem „!". Pisząc wyrażenia w linii poleceń możemy je umieścić w kilku wierszach. AutoCAD zlicza nawiasy i dopóki nie zostaną wszystkie zamknięte wyrażenie nie zostanie obliczone. W razie nie sparowania się jakiejś ilości nawiasów wyświetlany jest napis w postaci

```
((_>
```

informujący, ile nawiasów nie zostało zamkniętych (w tym przykładzie dwa).

Najwygodniejszy sposób użycia programu AutoLISP'a polega na umieszczeniu wyrażeń w pliku tekstowym z rozszerzeniem LSP, który można utworzyć np. systemowym notatnikiem Windows. Później plik ten można wczytać funkcją Lispa **load** lub poleceniem **WCZYTAJAPL**. Wczytanie funkcją Lispa wygląda np. tak:

```
(load "d:\\student\\acad\\test.lsp")
```

To spowoduje wczytanie i obliczenie wyrażeń z pliku **test.lsp** znajdującego się na dysku **D:** w katalogu **\student\acad**.

Skrypty - wsadowe przetwarzanie poleceń

Często dochodzi do sytuacji kiedy konstruktor wykonuje obliczenia jakimś programem zewnętrznym (np. Excel, MathCAD itp.) a potem musi według nich wykonać model geometryczny w AutoCAD'zie. W tym wypadku można skorzystać ze *skryptów*.

Skrypt jest plikiem tekstowym ASCII z rozszerzeniem SCR zawierającym polecenia AutoCAD'a wypisywane **dokładnie** tak samo jak w linii poleceń. Dokładnie oznacza, że każda spacja, każde wciśnięcie ENTER ma znaczenie. Jest to ważna uwaga bo nadmiarowe spacje, umieszczone zwłaszcza na końcu wiersza, są normalnie niewidoczne w edytorze a mają kolosalne znaczenie dla realizacji skryptu. Skrypt można utworzyć systemowym notatnikiem. Pamiętajmy, aby plik zapisać z rozszerzeniem SCR. Plik skryptu uruchamiamy poleceniem **pokaz**.

Skrypty stosujemy do realizacji zadań nieinteraktywnych tzn. nie wymagających ingerencji użytkownika. Poniżej przykład skryptu o nazwie **prost.scr** rysującego prostokąt o wymiarach 200 **x** 100 z narożnikiem w punkcie 0,0. Dla podkreślenia struktury pliku użyto znaczka ↵oznaczającego miejsca wciśnięcia klawisza ENTER

```
linia↵
0,0↵
@200,0↵
@0,100↵
@-200,0↵
z↵
```

Między znakami nie występuję żadna spacja. Wczytanie pliku o takiej treści poleceniem **pokaz** spowoduje narysowanie prostokąta.

Pisząc skrypty musimy znać na pamięć przebieg dialogu używanych poleceń. Jakiekolwiek błędy w skrypcie są trudne do wykrycia i powodują przerwanie wykonywania skryptu. Skrypty można też generować innymi programami. Szczególnie jeśli program taki piszemy osobiście korzystając z jakiegoś języka programowania np. Pascala.

Oto przykład jak utworzyć plik skryptu korzystając z Excel'a . Celem niech będzie narysowanie jednego okresu sinusoidy o amplitudzie 200 jednostek z dokładnością do 10º. W tym celu:

1. Otwórzmy Excela
2. Wpiszmy w kolumnie A liczby 0, 10, 20 .. 360 (komórki A1..A37). Będzie to kolumna rzędnych *x*.
3. Wpiszmy w komórce B1 wzór `=200*sin(A1*pi()/180)`
4. Wypełnijmy tym wzorem kolumnę B aż do pozycji B37 (kliknąć na komórce B1 i ciągnąć za jej prawy dolny narożnik ramki aż do B37). Będzie to kolumna odciętych *y*.
5. Z menu plik wybierzmy polecenia **Zapisz jako**
6. W oknie dialogowym wybierzmy typ **CSV (rozdzielany przecinkami) (*.csv)**
7. Odszukajmy katalog **Student** na dysku **D:**
8. Wpiszmy nazwę **sinus** i przycisk **OK**.
9. Teraz w programie Total Commander lub Mój Komputer odszukajmy plik **`sinus.csv`** i zmieńmy mu nazwę (właściwie tylko rozszerzenie) na **`sinus.scr`**.
10. Otwórzmy otrzymany plik notatnikiem lub klawiszem F4 w Total Commanderze.
11. Dopiszmy na początku wiersz z tekstem **plinia**
12. Dopiszmy pusty wiersz na końcu pliku (sam ENTER)
13. Zamieńmy (Ctrl-H) wszystkie przecinki na kropki a potem średniki na przecinki (kolejność wymiany jest ważna).
14. Zapiszmy plik i zamknijmy edytor.
15. W AutoCAD'zie wyłączmy tryb OBIEKT.
16. Wydajmy polecenie **pokaz** i wczytajmy plik **`sinus.scr`**
17. Wykonajmy plecenie **zoom zakres**.

Jeśli wykonaliśmy wszystko bezbłędnie na ekranie powinna się pojawić krzywa w kształcie sinusoidy. Tą dość długą procedurę można ominąć jeśli skorzysta się z Pascala[1]. Poniżej pokazano jak powinien wyglądać program realizujący to samo zadanie.

```
Program sinus;
uses math;
var
   y : real;
   f : text;
   nazwa : string;
   k1, k2, x : integer;
begin
{Inicjujemy dane}
  k1 := 0;
  k2 := 360;
  nazwa := 'sinus.scr';
  writeln(
    'Program generuje sinus od', k1,' do ',k2,
    ' stopni w formie skryptu dla AutoCAD');
{Otwierzmy plik}
```

[1] W pracowni jest zainstalowany Free Pascal, przy pomocy którego można utworzyć program

```
  assign(f, nazwa);
  rewrite(f);
{pierwszy wiersz pliku to polecenie rys. polilini po ang.}
  writeln(f,'_pline');
{Kolejne wiersze od współrzędne punktow w przedziale od k1 do k2 }
  for x:=k1 to k2 do
    begin
     {obliczmy y dla zadanego x przeliczonego ze stopni na radiany}
      y := 200.0*sin(pi*x/180.0);
     {wpisujemy do pliku pare x,y}
      writeln(f,x,',',y:0:4);
    end;
 {pusty wiersz konczacy polecenie plinia}
  writeln(f);
{zamykamy plik}
  close(f);
end.
```

Ten program generuje sinusa z dokładnością do 1º. Możliwe zmiany w tym programie do zorganizowanie wczytania katów *k1* i *k2* oraz nazwy pliku *nazwa*.

Aby skrypty były uniwersalne należy stosować w nich angielskie nazwy poleceń i opcji. Dzięki temu możne je wczytywać AutoCAD'em dla dowolnej wersji językowej.

Wykaz poleceń

Polecenie	Opis
kalk, _cal, _cal, 'kalk, 'cal	Wywołanie kalkulatora. Wywołanie nakładkowe 'cal lub 'kalk umożliwia użycie wyników wyrażeń jako odpowiedzi na pytania AutoCAD'a
pokaz, _script **Narzędzia - Pokaz**	Pozwala wczytać i uruchomić skrypt. Polecenie można wydawać nakładkowo.

Legenda: – linia poleceń; **M:** – menu; – pasek narzędziowy

Ćwiczenie nr 14 – Zadania do wykonania

Zadanie A Kalkulator

1. Używając kalkulator zdefiniuj następujące symbole: *w*=10.65/2, *r*=24.33, *h*=2π*r* oraz **p** jako punkt odległy od punktu (10,20) o *h* jednostek w poziomie i *w* jedn. w pionie
2. Korzystając ze zdefiniowanych zmiennych i kalkulatora
   Narysuj okrąg o środku w punkcie **p** i o promieniu *r*.
   Narysuj prostokąt o szerokości *w* i wysokości *h*.
3. Narysuj dowolny okrąg. Teraz korzystając z kalkulatora narysuj inny okrąg o polu równym połowie pola okręgu poprzedniego. Wykorzystaj funkcje **`rad`**.
4. Narysuj okrąg o polu 314.15

Zadanie B Autolisp

1. Przetestuj przykłady podane w poprzednim rozdziale oraz funkcje wymienione w tabeli.
2. Oblicz przy pomocy AutoLISP'a następujące wyrażenia:
   (12.4+45.6)(17.33 – 5.32)
   πa/180 gdzie a=23.565 (zamiana stopni na radiany)
   πr^2 gdzie r=23.4
3. Z dwóch ostatnich wyrażeń uczyń definicje funkcji o nazwie DEG i POLE. Zapisz je w pliku z rozszerzeniem LSP. Wczytaj go i przetestuj.
4. Zapisz zdefiniowane w poprzednim rozdziale funkcje w pliku z rozszerzeniem LSP. Wczytaj go i przetestuj.
5. Zrób makro do pomiaru kąta. Wykorzystaj makro zapisane w poprzednim rozdziale.

Zadanie C Automatyzacja poleceń

1. Posługując się załączonymi przykładami utwórz krzywą o wzorze y=1/x w zakresie 0.01-100 rysowaną poleceniem plinia
2. Wyznacz pole ograniczone liniami y=1/x, x=1, y=0 i x=100
   a. po narysowaniu krzywej z pkt 1 narysuj dodatkowe linie poziomą y=0 oraz x=1 i x=100
   b. utnij linie wychodzące poza wyznaczane pole
   c. przekształć pozostałe po obcięciu obiekty w region poleceniem **REGION**
   d. Poleceniem **POLE** z opcją **Obiekt** wyznacz pole utworzonego regionu