Instrukcja `input`
W zaprogramowaniu kolejnych zadań przyda się umiejętność wprowadzania wartości 'z zewnątrz' do programu (wczytywanie danych). Na przykład:
`liczba = input("Podaj liczbe:")`
Działanie instrukcji `input`:
Wypisany zostanie komunikat: `Podaj liczbe:`
Należy wpisać wartość, nacisnąć na Enter
Wprowadzona wartość zostanie podstawiona na zmienną `liczba`.

Małe zadania:
- dla danej wartości promienia `r` wyznaczyć pole powierzchni oraz objętość kuli;
- dla danej wartości `a` oznaczającej bok sześcianu, obliczyć pole powierzchni, obwód oraz długość przekątnej sześcianu;
- dla dwóch wartości oznaczonych `a` i `b` obliczyć pole i obwód prostokąta oraz długość jego przekątnej;
- dla dwóch wartości `r` i `h` obliczyć pole powierzchni oraz objętość cylindra o promieniu podstawy `r` oraz wysokości `h`.
- dla danej wartości promienia `r` wyznaczyć pole powierzchni trójkąta równobocznego wpisanego w okrąg o takim promieniu.
- dla danej wartości promienia `r` wyznaczyć pole powierzchni kwadratu wpisanego w okrąg o takim promieniu.
- dla danej wartości promienia `r` wyznaczyć pole powierzchni pięciokąta foremnego wpisanego w okrąg o takim promieniu. Uogólnić program na przypadek dowolnego wielokąta foremnego (potraktować liczbę boków jako dodatkową daną).

Zadanie:
Napisać program, który 'odgadnie' liczbę naturalną z przedziału [1, 50] wylosowaną przez komputer. Można zastosować różne algorytmy.

Instrukcja **Select**
Instrukcja select umożliwia wykonywanie różnych czynności, w zależności od wartości przyjmowanej przez wrażenie. Można ją uznać za bliską 'krewną' instrukcji warunkowej `if-then-elseif`.
Instrukcja select ma postać:

```
select wyrażenie
      case wartość1
      instrukcje1
      case wartość2
      instrukcje2
.....
      case wartość_n
      instrukcje_n
else
      instrukcje
end
```

Działanie jest następujące:
wyznaczana jest wartość wyrażenia '`wyrażenie`'. Jeśli `wyrażenie=wartość1` to wykonywane są `instrukcje1`, jeśli `wyrażenie=wartość2`, wykonywane są `instrukcje2`, i tak dalej.

Instrukcje znajdujące się po frazie `else` są wykonywane jeśli `wyrażenie` nie jest równe żadnej z wcześniej wymienionych wartości.

Na przykład:

```
i= ....;
select i
case 1
disp ("jeden")
case 2
disp ("dwa")
case 3
disp (" trzy ")
else
error ( " nieoczekiwana wartość" )
end
```

Operacje wejścia/wyjścia

Podczas wykonywania programu niezbędna jest komunikacja z komputerem: z jednej strony trzeba mieć możliwość wprowadzania danych, równie ważna jest też możliwość wypisania i zachowania wyników. Na marginesie zwracamy uwagę, że wyniki mogą przyjmować formę graficzną, w postaci np. wykresów funkcji, teraz jednak koncentrujemy się na wynikach zapisywanych w postaci tekstowej.

Kolejno omówimy komunikację bezpośrednią (interaktywną) pomiędzy programistą (lub użytkownikiem) a programem, korzystanie z możliwości odczytu oraz zapisania do pliku zostanie omówione w dalszej części opracowania.

Wprowadzanie danych

Rozróżnia się wprowadzanie danych liczbowych (numerycznych) i tekstowych, w obydwu przypadkach wykorzystuje się funkcję `input`. Często na wprowadzanie danych mówi się 'czytanie danych'.

Wprowadzanie wartości liczbowych

```
zmienna = input ('Tekst');
```

Działanie jest następujące: na konsoli wyświetla się `Tekst`, a działanie programu jest wstrzymane w oczekiwaniu na wprowadzenie wartości. Wprowadzona wartość (należy nacisnąć na Enter) jest podstawiana pod zmienną `zmienna`. Jak łatwo zauważyć, `Tekst` pełni rolę informacyjną. Na przykład, aby obliczyć sumę `n` kolejnych liczb naturalnych, można wykorzystać następujący ciąg instrukcji:

```
k=0;
n=input('Największa sumowana liczba naturalna')
for i=1:n
      k=k+i;
end;
```

Realizacja tego fragmentu programu przebiega następująco: Zmiennej `k` zostaje nadana wartość 0. Następnie na konsoli pojawia się komunikat `Największa sumowana liczba naturalna`, i wykonanie obliczeń jest zawieszone do chwili wprowadzenia (jakiejś) wartości. Wprowadzona wartość zostaje podstawiona pod zmienną `n` i rozpoczyna się wykonywanie kolejnych instrukcji.

Dygresja: Sumowanie ciągu wartości
Przyjrzyjmy się jeszcze schematowi sumowania ciągu wartości, w tym przypadku są to kolejne liczby naturalne. Kluczową rolę pełni tu instrukcja `k=k+i`, wykonywana w pętli. Przypomnijmy, że instrukcję tę należy odczytać jako: do wartości zmiennej `k` dodaj wartość zmiennej `i`, a wynik podstaw pod zmienną `k` (niszcząc jej poprzednią wartość). Zauważmy też, że zmienna `i` jest zmienną sterującą pętli, a jej wartość zmienia się w każdym obiegu pętli o jeden, począwszy od 1 aż do `n`.
Załóżmy, że jako `n` wprowadzono 5, a zatem wyznaczamy sumę liczb od 1 do 5.

- Pierwsze wykonanie pętli, `i=1: k=k+1`, czyli `k=0+1=1`

Zauważmy, jak ważne było nadanie zmiennej `k` wartości 0 przed rozpoczęciem sumowania. Można to porównać do wyczyszczenia stołu, na który będziemy dokładać (dodawać) kolejne sumowane wartości.

- Drugie wykonanie pętli, `i=2: k=k+2`, czyli `k=1+2=3`
- Trzecie wykonanie pętli, `i=3, k=k+3`, czyli `k=3+3=6`
- Czwarte wykonanie pętli, `i=4, k=k+4`, czyli `k=6+4=10`
- Piąte wykonanie pętli, `i=5, k=k+5`, czyli `k=10+5=15`

W tym miejscu działanie programu się kończy, bowiem `i` osiągnęło wartość końcową `n`.

Wprowadzanie wartości znakowych
Podobnie odbywa się wprowadzanie wartości znakowych.
Instrukcja `input` ma postać:
```
imie=input("Jak masz na imie? ","string")
```
W wyniku wykonania tej instrukcji na konsoli pojawi się napis „`Jak masz na imię`”, a po wprowadzeniu odpowiedzi zostanie ona podstawiona pod zmienną `imie`.
Być może rozróżnienie pomiędzy zmiennymi liczbowymi i znakowymi wydaje się Wam teraz czymś nienaturalnym. Niech pewnym wyjaśnieniem będzie konieczność rozróżnienia pomiędzy zmiennymi, które mają wartość liczbową (i w odniesieniu do których można wykonywać operacje arytmetyczne) oraz zmiennymi znakowymi, przechowującymi ‘napisy’.

Wypisanie wyników na konsoli
Do wypisania wyników na konsoli Scilaba służy funkcja `disp`, na przykład `disp(k)` spowoduje wyświetlenie na konsoli wartości zmiennej `k`.
Uwagi o `disp`:

- Można wypisać równocześnie wartości więcej niż jednej zmiennej, na przykład `disp(i,j,k)`. Zmienne zostaną wyświetlone w odwrotnej kolejności i w kolejnych wierszach.
- Czytelniejszy i bardziej zwarty wydruk otrzyma się przekształcając wartości liczbowe na łańcuchy znakowe w następujący sposób:

```
disp (string(i)+string(j)+string(k))
```

- Przydałyby się spacje, żeby zwiększyć czytelność ... a zatem trzeba je dodać:

```
disp (string(i)+' '+string(j)+' '+string(k))
```

- Zauważmy, że w odniesieniu do zmiennych/stałych znakowych, operator + oznacza dodawanie rozumiane jako łączenie (‘sklejanie’) tekstów. Inaczej na taką operację mówi się *konkatenacja*.
- Dodatkowo, można wprowadzić opisy, na przykład takie:

```
disp('i='+string(i)+' j='+string(j)+' k='+string(k))
```

Uwagi te przedstawia poniższy zapis z konsoli Scilaba:
```
-->i=1;
-->j=2;
```

```
 -->k=3;
 -->disp(i,j,k)
     3.
     2.
     1.
-->disp (string(i)+string(j)+string(k))
  123
 -->disp (string(i)+' '+string(j)+' '+string(k))
  1 2 3
 -->disp('i='+string(i)+' j='+string(j)+' k='+string(k))
  i=1 j=2 k=3
```

Bezpośrednia komunikacja z komputerem podczas wykonywania programu jest cenna, ale w wielu przypadkach niewystarczająca. W szczególności w sytuacjach, gdy program korzysta z dużej liczby danych oraz gdy generuje duże zestawy wyników, niezbędna staje się umiejętność komunikacji z plikami.

Laboratorium – trochę całkowania, trochę rysowania

1. Narysować wykres funkcji:

$$y = ax^2 + bx + c$$

w przedziale [-10, 10] z krokiem Δx=0.1 dla a=-0.2, b=0.5, c=10.

2. Wyznaczyć punkty przecięcia z osią OX (oznaczenie x1 i x2).

3. Korzystając z metody prostokątów, wyznaczyć pole ograniczone wykresem i osią OX między punktami x1 i x2.

a) Jak zmienia się wartość całki w zależności od kroku całkowania numerycznego?

b) Naszkicować 'wizualizację' metody prostokątów korzystając z funkcji **xrects**(m). Funkcja ta rysuje ciąg prostokątów zdefiniowanych poprzez macierz m, gdzie m =[x; y; w; h] (uwaga na średniki). Oznaczenia: x,y – współrzędne lewego górnego rogu prostokątów, w – szerokości rysowanych prostokątów, h – wysokości.
Dodatkowo funkcja xrects może być uzupełniona o parametr definiujący kolor wykreślania.

c). Porównać graficznie dokładność metody w zależności od kroku całkowania.

```
clf()
a=-0.2;
b=0.5;
c=10;
x=(-10:0.1:10);
y=a*x^2+b*x+c;
plot (x,y)
xgrid()
delta=b^2-4*a*c;
x1=(-b-sqrt(delta))/(2*a)
x2=(-b+sqrt(delta))/(2*a)
/// UWAGA! Czy x1 < x2?
xp=(x2:0.5:x1);
yp=a*xp^2+b*xp+c;
w=0.5*ones(xp);
m=[xp;yp;w;yp];
// xp, yp - lewy górny róg
// w - szerokość prostokąta
// ostatni argument - wysokość
xrects(m,15*ones(xp))
// kolor wypelnienia
```

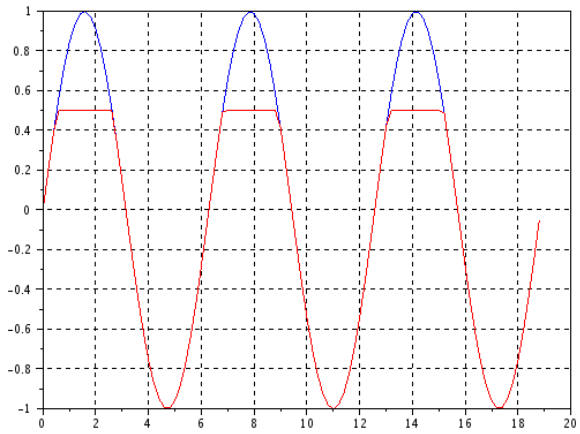
```
// 5- czerwony
// 10-niebieski
// 15 zielony
// 20 - czerwony
//-15 - brak wypełnienia, zielone krawędzie
// -5 brak wypełnienia, czerwone krawędzie
```

4. Dla dowolnych podanych wartości parametrów a, b i c (funkcja **input**) definiujących funkcję kwadratową

$y = ax^2 + bx + c$

stwierdzić, czy przecina ona oś OX.
Napisać program, który w zależności od sytuacji (**IF**...) wypisze odpowiedni komunikat: przecina w dwóch punktach, nie przecina (funkcja **disp**).
Jeśli wykres przecina oś OX w dwóch punktach, wyznaczyć je i wypisać ich wartości.

5. Narysować wykres funkcji *y=sin x* w przedziale [0, 6π].
a) Wyznaczyć przybliżoną pochodną i narysować jej wykres.
b) Narysować wykres funkcji powstałej poprzez zastąpienie wszystkich wartości większych niż 0.5 wartością 0.5 (patrz rysunek).



c) Narysować wykres funkcji powstałej poprzez zastąpienie wszystkich wartości większych niż 0.5 wartością 0.5 oraz mniejszych niż -0.5 wartością -0.5.
d) Jak w sposób przybliżony można obliczyć pole 'obciętej' części wykresu?
Proponowane narzędzia: całkowanie numeryczne, może funkcja
`[buttons,x,y]=xclick()`....

6. Obliczyć długość łuku wyznaczonego przez wykres funkcji y=sin(x) w przedziale [0, π] korzystając ze wzoru na długość łuku:

$$L = \int_{x1}^{x2} \sqrt{1 + f'(x)^2}\, dx$$

Dodatkowo narysować wykres funkcji

$$y = \sqrt{1 + f'(x)^2}$$

SCILAB

Materiały opracowała Anna Trykozko, we współpracy z Łukaszem Czerwińskim