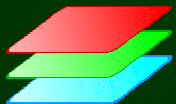




Języki skryptowe w programie Plans

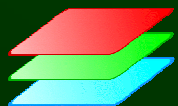
Tomasz Zdun



Warsztaty użytkowników programu PLANS – Kościelisko 2010

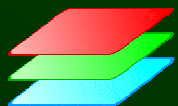
Zalety skryptów

- Automatyzacja powtarzających się czynności
- Rozszerzenie możliwości programu
 - Budowa własnych algorytmów analizy
 - Współpraca z innymi systemami(np. MS Office, system SCADA)
 - Współpraca wielu instancji programu Plans
 - itd...



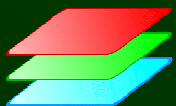
Języki skryptowe w Plansie

- język makropoleceń JMP
- VBScript
- JScript



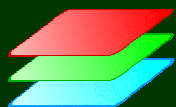
Język makropolecień JMP

- Stworzony na potrzeby programu Plans
- Dostęp do wszystkich funkcji programu Plans
- Ograniczona funkcjonalność:
 - brak tablic
 - brak zmiennych lokalnych
 - ograniczona możliwość przekazywania wartości do procedur
 - brak funkcji



JScript i VBScript

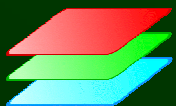
- Języki „wbudowane” do systemu Windows
- Wykorzystywane w wielu aplikacjach oraz na stronach WWW
- Bogata składnia:
 - pętle i instrukcje warunkowe
 - tablice i obiekty
 - funkcje
 - obsługa wyjątków
- Bogata dokumentacja



VBscript

- Oparty na języku BASIC
- Składnia podobna do Visual Basic for Application (pakiet MS Office), ale NIE taka sama

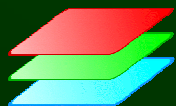
```
dim i, j
for i=1 to 10
    j = j + i
next
cprintf(j)
```



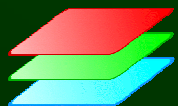
JScript

- Wersja języka JavaScript opracowana przez Microsoft (Internet Explorer)
- Składnia podobna do języków Java, C i C++

```
var i, j=0;  
for(i=1;i<10;i++)  
{  
    j += i;  
}  
cprintf(j);
```

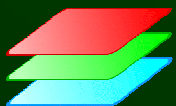


SKŁADNIA JĘZYKA JSCRIPT. PODSTAWY

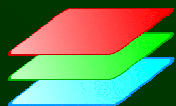


Podstawowe zasady

- Instrukcje oddzielone średnikiem lub końcem wiersza (elastyczność)
- Ważna wielkość liter
- Komentarz w jednej linii:
`//komentarz`
- Komentarz w wielu liniach:
`/* komentarz
komentarz */`
- Blok kilku instrukcji wzięty w nawiasy klamrowe `{}`
- Wyświetlenie tekstu na okno: funkcja `cprintf(text)`



ZMIENNE



Deklaracja i typ

- Pierwsze użycie jest również deklaracją zmiennej

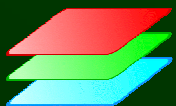
```
x = 10;
```

- Brak typów zmiennych

```
x = 10;
```

```
x = "Zmienna tekstowa";
```

```
x = new Array();
```



Zakres ważności

- Zmienne globalne:

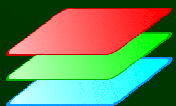
```
x = 10;  
var x = 10;
```

- Zmienne lokalne funkcji - var:

```
function f1()  
{  
    var x = "tekst";  
}
```

- Zalecane deklarowanie wszystkich zmiennych ze słowem

var



Konwersja typów

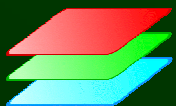
- Typ numeryczny → tekst
 - konwersja automatyczna

```
cprintf("Liczba" + 20);
```
 - konwersja jawna

```
cprintf("Liczba" + x.toString(10));
```
- Tekst → typ numeryczny
 - konwersja jawna do liczby całkowitej

```
x = 10 + parseInt("20");
```
 - do liczby zmiennoprzecinkowej

```
x = 10.8 + parseFloat("20.9");
```



Formatowanie liczb rzeczywistych

```
var x = 100/3;
```

- Konwersja do tekstu:

```
x.toString(10)
```

(10-system dziesiętny, 2-system binarny itp.)

- Zadana liczba cyfr po kropce

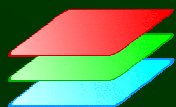
```
x.toString(ile)
```

- Zadana liczba cyfr znaczących (przed i po kropce)

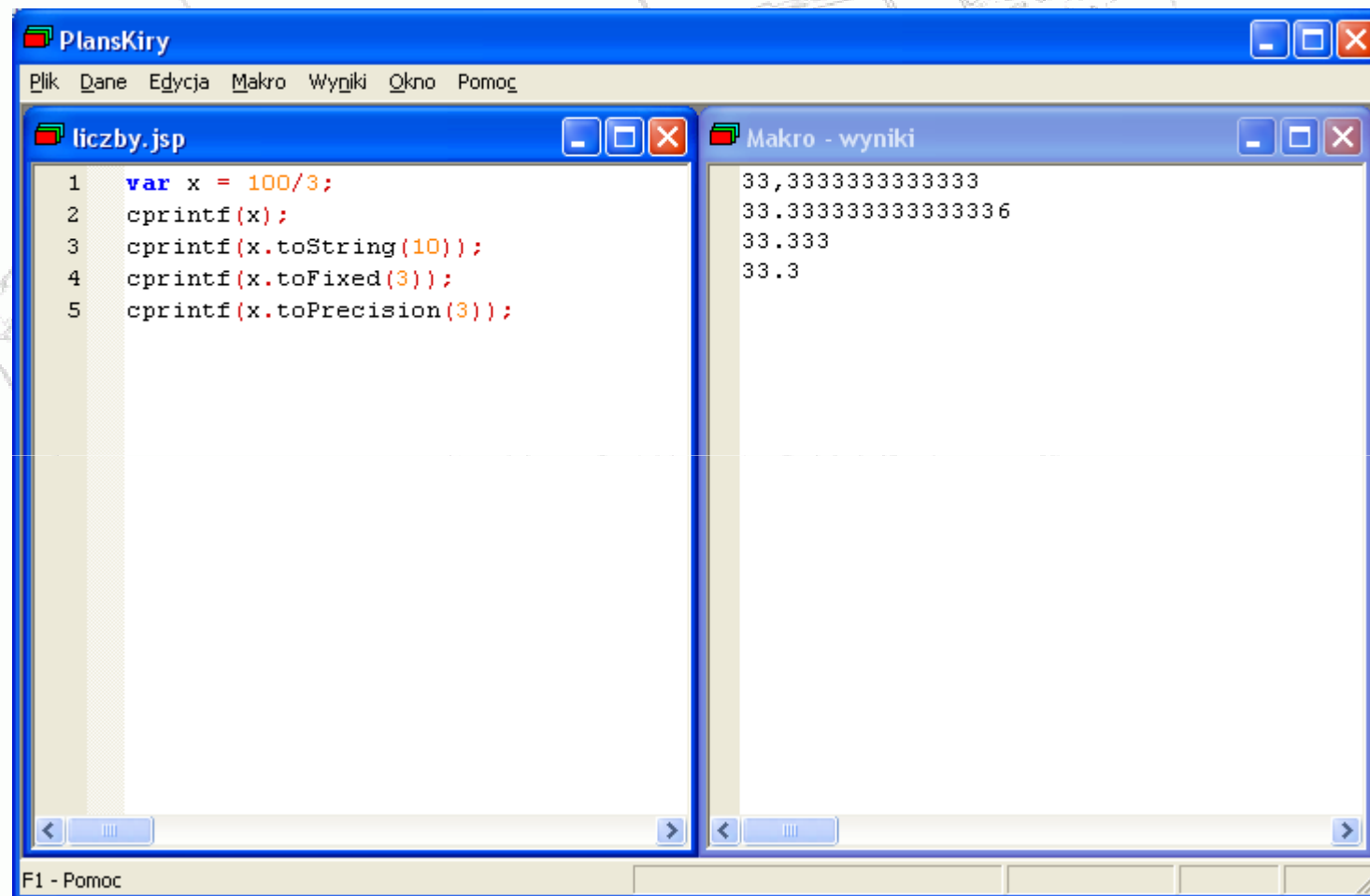
```
x.toFixed(ile)
```

- Zadana dokładność w postaci długość.precyzja:

brak



Formatowanie liczb rzeczywistych



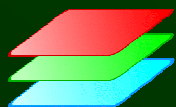
The screenshot shows the PlansKiry application window. It has a menu bar with 'Plik', 'Dane', 'Edycja', 'Makro', 'Wyniki', 'Okno', and 'Pomoc'. There are two main panes. The left pane, titled 'liczby.jsp', contains the following code:

```
1 var x = 100/3;  
2 cprintf(x);  
3 cprintf(x.toString(10));  
4 cprintf(x.toFixed(3));  
5 cprintf(x.toPrecision(3));
```

The right pane, titled 'Makro - wyniki', displays the output of the macro execution:

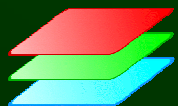
```
33,33333333333333  
33.333333333333336  
33.333  
33.3
```

At the bottom of the window, there is a status bar with 'F1 - Pomoc' and some navigation buttons.





INSTRUKCJE STERUJĄCE



Wyrażenia logiczne

Prawda	true
--------	------

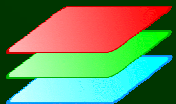
Fałsz	false
-------	-------

Porównania:	==, !=, <, >, >=, <=
-------------	----------------------

Iloczyn logiczna (and)	&&
------------------------	----

Suma logiczna (or)	
--------------------	--

Negacja	!
---------	---



Wyrażenia logiczne

- x w przedziale (10,20)

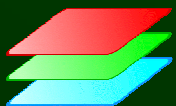
`x > 10 && x < 20`

- x z poza przedziału (10,20)

`x <= 10 || x >= 20`

- x w przedziale (0,10) lub (20,30)

`(x > 0 && x < 10) || (x > 20 && x < 30)`



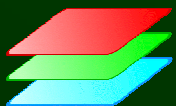
Instrukcja warunkowa if

- Postać 1

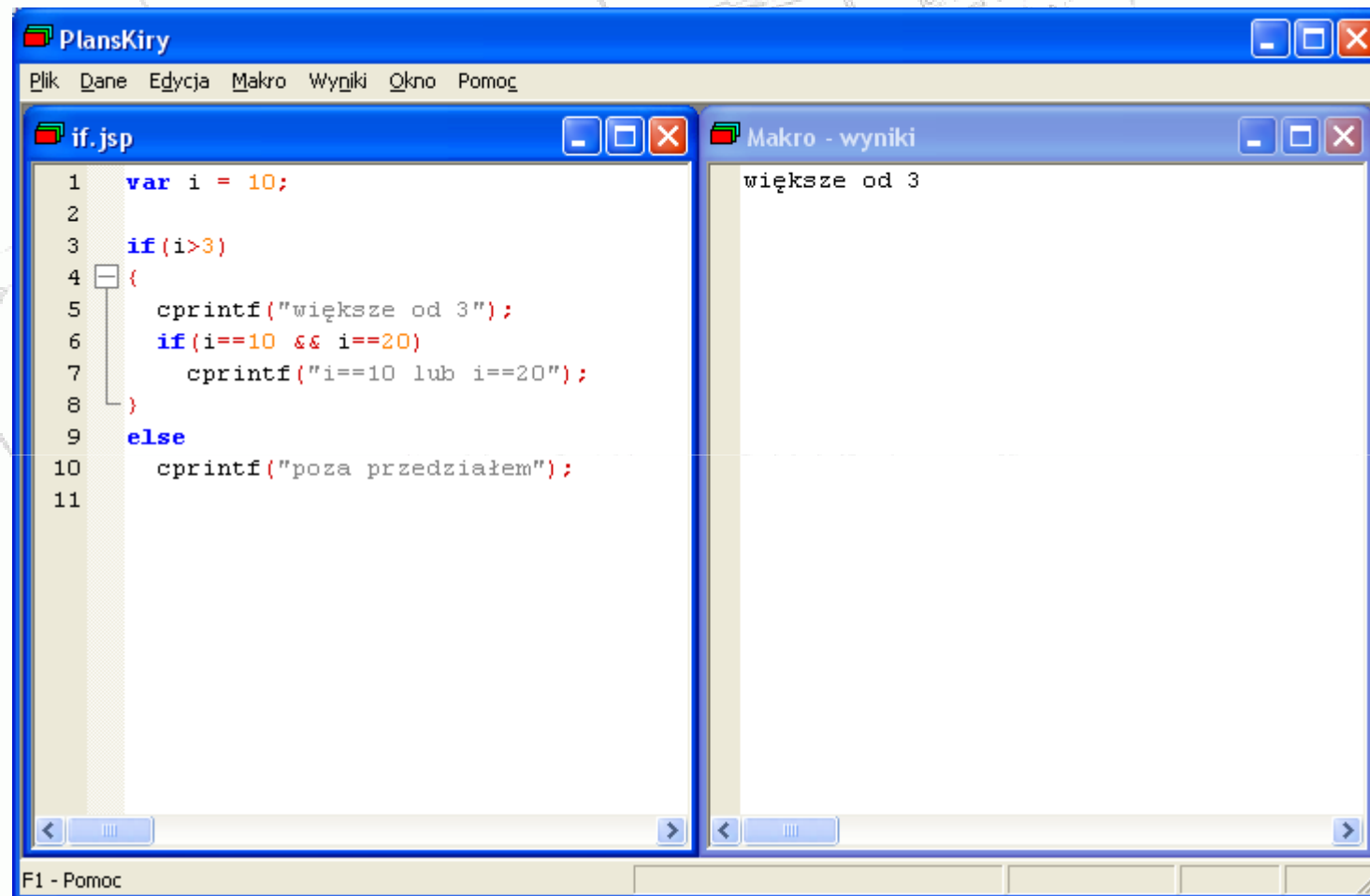
```
if(warunek)instrukcja1;  
else instrukcja2;
```

- Postać 2

```
if(warunek)  
{  
    instrukcja1;  
    instrukcja2;  
}  
else ...
```



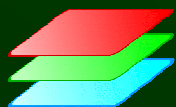
Instrukcja warunkowa if



The screenshot shows the PlansKiry application interface. The main window is titled 'PlansKiry' and contains a menu bar with 'Plik', 'Dane', 'Edycja', 'Makro', 'Wyniki', 'Okno', and 'Pomoc'. Below the menu bar, there are two panes. The left pane, titled 'if.jsp', contains the following code:

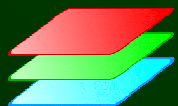
```
1  var i = 10;  
2  
3  if(i>3)  
4  {  
5      cprintf("większe od 3");  
6      if(i==10 && i==20)  
7          cprintf("i==10 lub i==20");  
8  }  
9  else  
10     cprintf("poza przedziałem");  
11
```

The right pane, titled 'Makro - wyniki', displays the output of the code: 'większe od 3'. The status bar at the bottom of the application window shows 'F1 - Pomoc'.

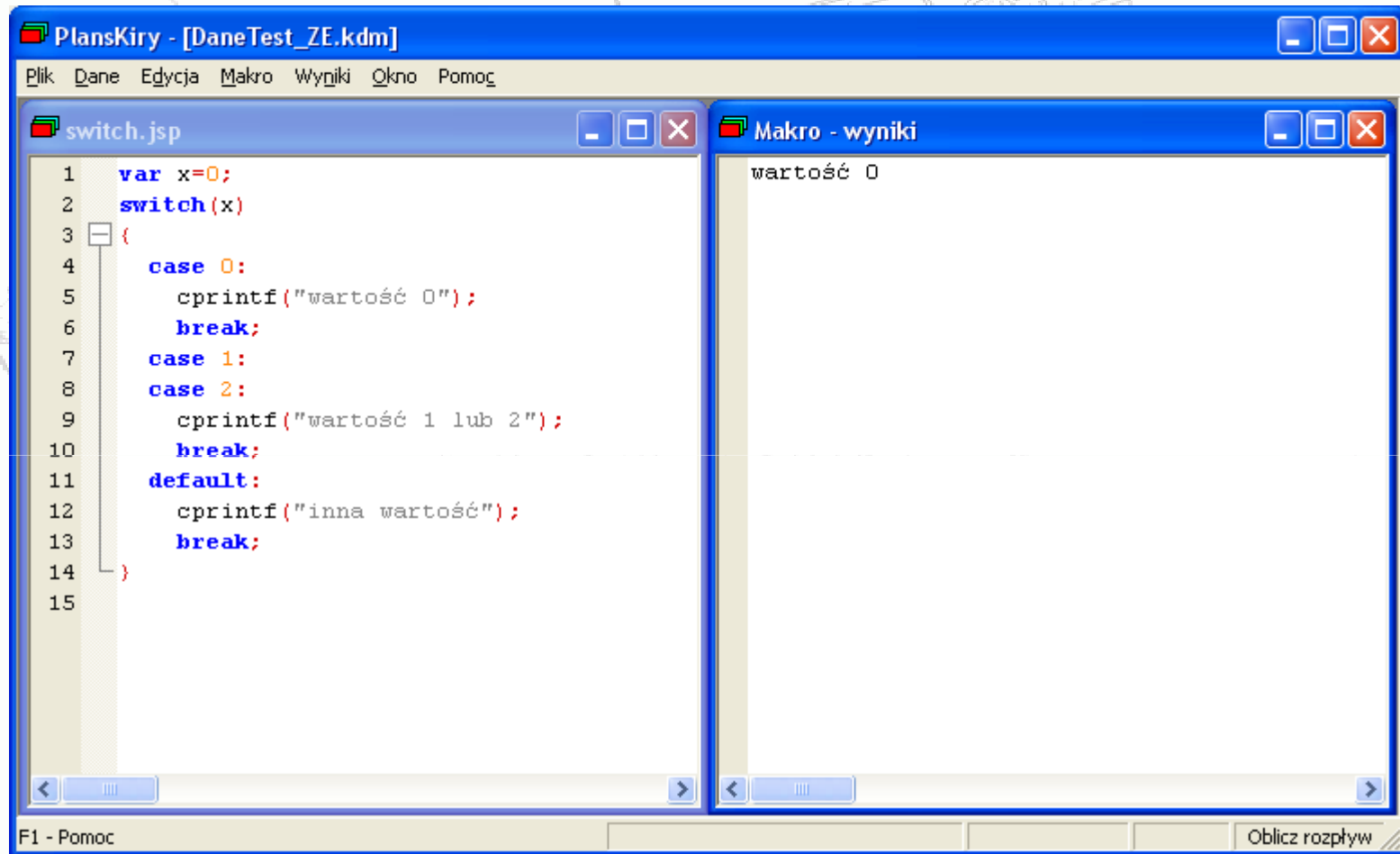


Instrukcja wyboru switch

```
switch(wyrażenie){  
    case wartość1:  
        instrukcja1;  
        break;  
    case wartość2:  
    case wartość4:  
        instrukcja2_3;  
        break;  
    default:  
        instrukcje3;  
        break;  
}
```



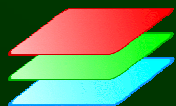
Instrukcja wyboru switch



The screenshot displays the PlansKiry IDE interface. The main window, titled "PlansKiry - [DaneTest_ZE.kdm]", contains a menu bar with "Plik", "Dane", "Edycja", "Makro", "Wyniki", "Okno", and "Pomoc". Below the menu is a toolbar. The main editing area shows a file named "switch.jsp" with the following code:

```
1  var x=0;
2  switch(x)
3  {
4      case 0:
5          cprintf("wartość 0");
6          break;
7      case 1:
8      case 2:
9          cprintf("wartość 1 lub 2");
10         break;
11     default:
12         cprintf("inna wartość");
13         break;
14 }
15
```

To the right of the main editor is a smaller window titled "Makro - wyniki" (Macro - results). It displays the output of the macro execution: "wartość 0". At the bottom of the IDE, there is a status bar with "F1 - Pomoc" on the left and "Oblicz rozptyw" (Calculate spread) on the right.



Pętla for

```
for(inicjalizacja; warunek; inkrementacja)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

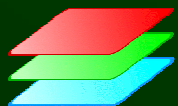
Inkrementacja

Zwiększenie licznika *i* o jeden


`i = i+1` \leftrightarrow `i+=1` \leftrightarrow `i++`

Zmniejszenie licznika *i* o jeden

`i = i-1` \leftrightarrow `i-=1` \leftrightarrow `i--`



Pętla for



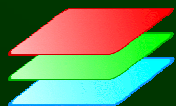
The screenshot shows the PlansKiry IDE with a project named [DaneTest_ZE.kdm]. The main editor window displays a file named `for.jsp` with the following code:

```
1  for( i=1; i<=10; i++)
2  {
3      cprintf(i+ '...');
4  }
5
6  for( i=10; i>0; i--)
7  {
8      cprintf('...' + i);
9  }
10
11
```

The right-hand pane, titled "Makro - wyniki", shows the output of the code execution:

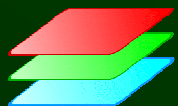
```
1...
2...
3...
4...
5...
6...
7...
8...
9...
10...
...10
...9
...8
...7
...6
...5
...4
...3
...2
...1
```

The status bar at the bottom indicates "F1 - Pomoc", "X=273, Y=76", and "Krok=12".



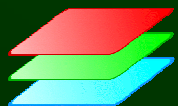
Pętla while

```
while(warunek)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

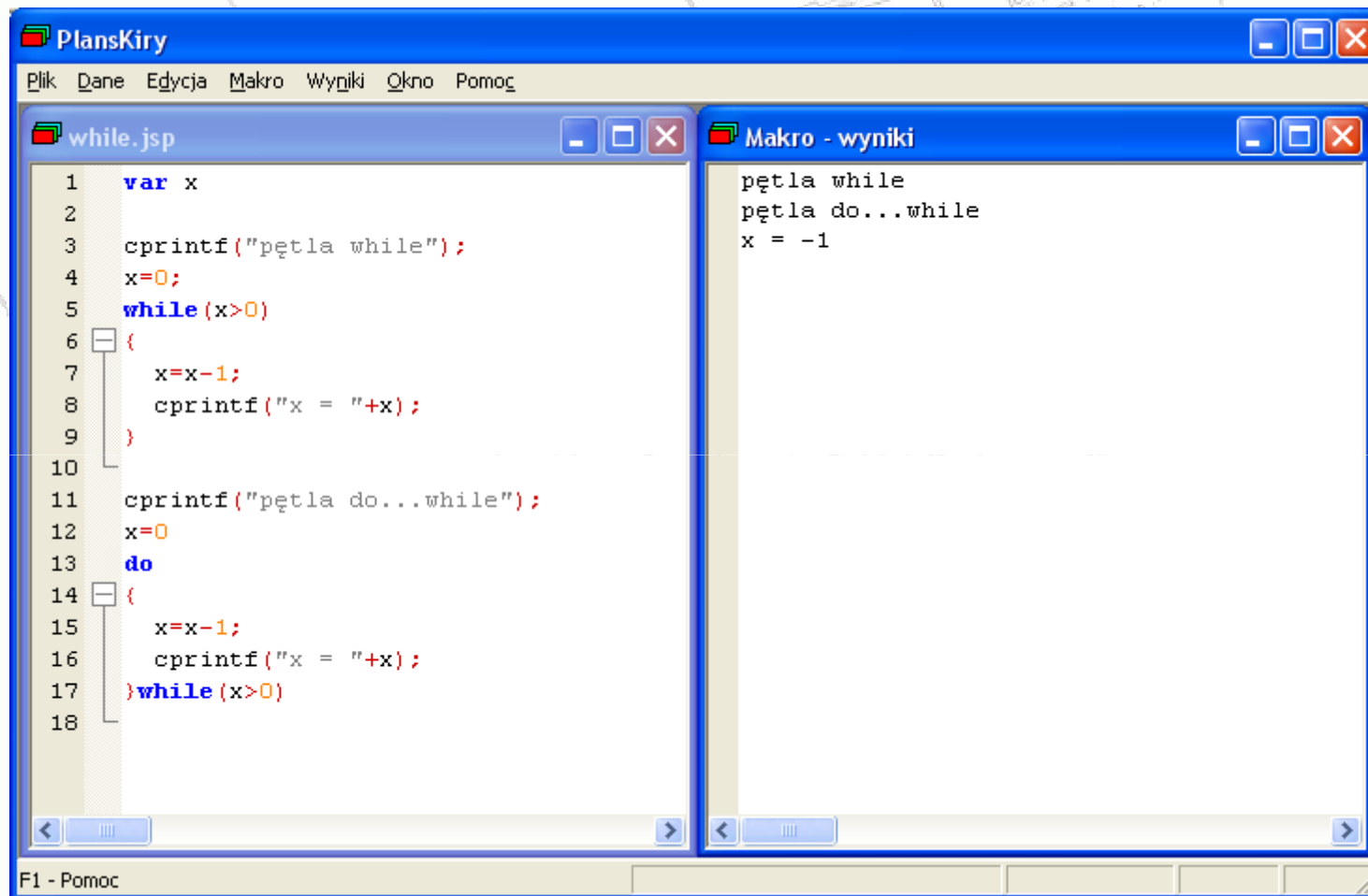


Pętla do...while

```
do
{
    instrukcja1;
    instrukcja2;
    ...
}
while(warunek)
```



Pętla while oraz do...while

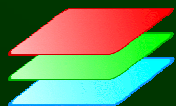


The screenshot shows the PlansKiry IDE with two windows. The main window, titled 'while.jsp', contains a C script with the following code:

```
1  var x
2
3  cprintf("pętla while");
4  x=0;
5  while(x>0)
6  {
7      x=x-1;
8      cprintf("x = "+x);
9  }
10
11 cprintf("pętla do...while");
12 x=0
13 do
14 {
15     x=x-1;
16     cprintf("x = "+x);
17 }while(x>0)
18
```

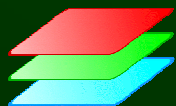
The second window, titled 'Makro - wyniki', shows the output of the script:

```
pętla while
pętla do...while
x = -1
```

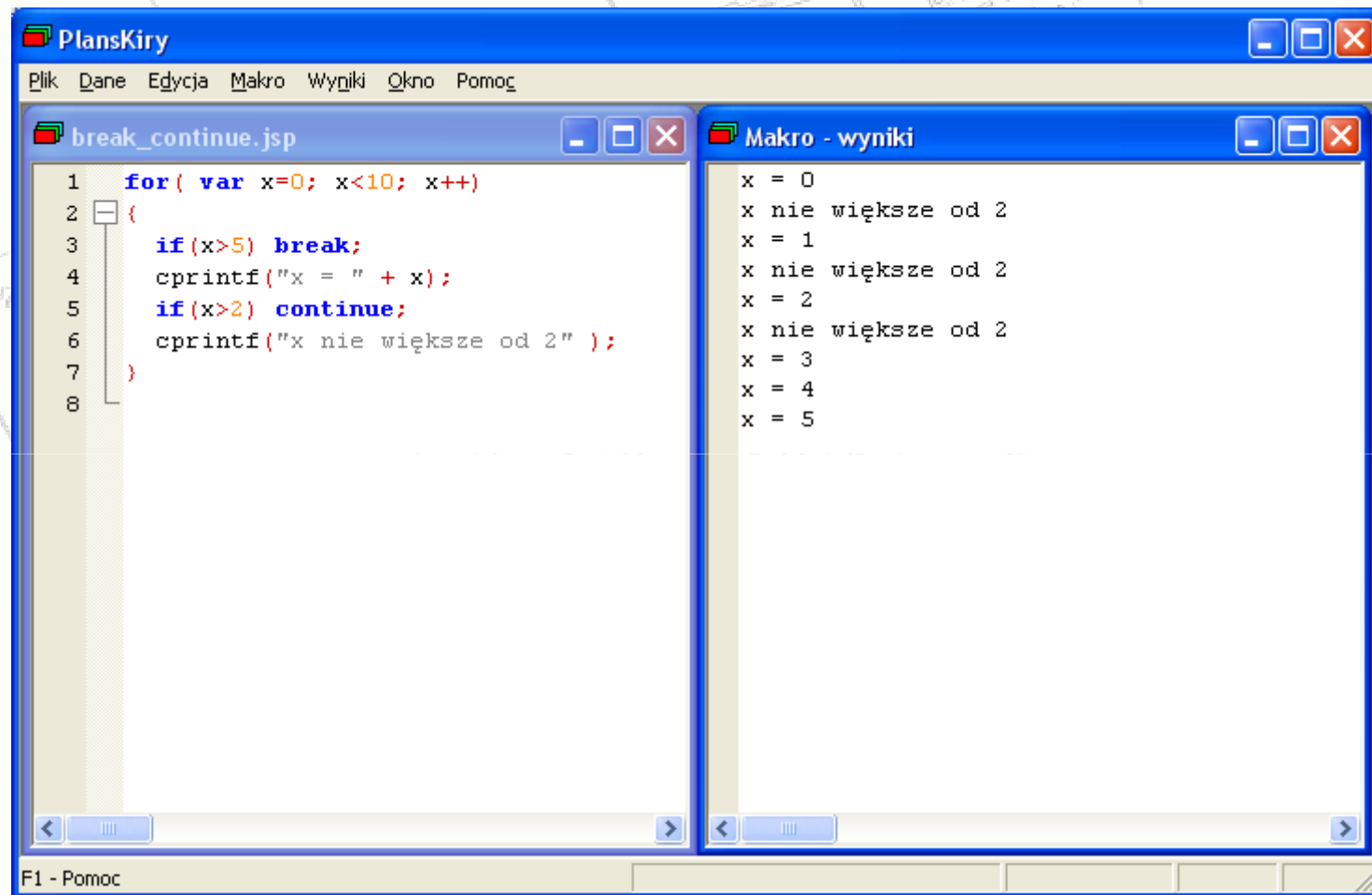


Instrukcje break i continue

- **break** – przerwanie wykonywania pętli
- **continue** – pominięcie dalszych instrukcji i sprawdzenie warunku pętli



Instrukcje break i continue



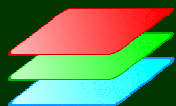
The screenshot shows the PlansKiry IDE with two windows. The main window, titled 'break_continue.jsp', contains the following JavaScript code:

```
1  for( var x=0; x<10; x++)
2  {
3      if(x>5) break;
4      cprintf("x = " + x);
5      if(x>2) continue;
6      cprintf("x nie większe od 2" );
7  }
8
```

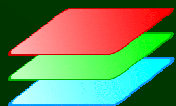
The second window, titled 'Makro - wyniki', displays the output of the code execution:

```
x = 0
x nie większe od 2
x = 1
x nie większe od 2
x = 2
x nie większe od 2
x = 3
x = 4
x = 5
```

The IDE interface includes a menu bar with 'Plik', 'Dane', 'Edycja', 'Makro', 'Wyniki', 'Okno', and 'Pomoc'. The status bar at the bottom shows 'F1 - Pomoc'.

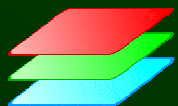


FUNKCJE



Definicje funkcji

```
function nazwa_funkcji(arg1, arg2,...)
{
    instrukcja1;
    instrukcja2;
    ...
    return(wartość);
}
```



Definicje funkcji - przykład

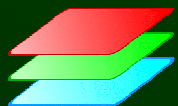
```
function SumaX2Y2(x, y)
{
    var z;
    z = x*x + y*y;
    return(z);
}
```



Wywołanie funkcji

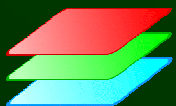
```
nazwa_funkcji(wartość1, wartość2,...)
```

wartość1 ... - przekazywane parametry do funkcji

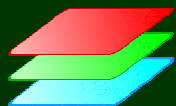


Wywołanie funkcji

```
function SumaX2Y2(x, y)
{
    var z;
    z = x*x + y*y;
    return(z);
}
var g = 20;
var h = SumaX2Y2(5,g);
```



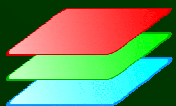
PRACA Z PLIKAMI



Obiekt FileSystemObject (FSO)

- Obiekt FSO jest częścią języka JScript
- Obiekt FSO udostępnia wiele metod do zarządzania folderami i plikami
- Dostęp do metod: *obiekt.metoda*
- Tworzenie obiektu FSO

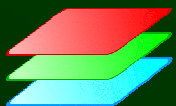
```
var fso = new ActiveXObject(  
    "Scripting.FileSystemObject");
```



Otwarcie pliku

```
var plik = fso.OpenTextFile( "nazwa.txt",  
    tryb, tworzyc );
```

- fso – utworzony obiekt FSO
- tryb:
 - 1 – do odczytu
 - 2 – do zapisu
 - 3 – do dopisywania na końcu pliku
- tworzyc – czy utworzyć plik, jeśli nie istnieje (true/false, domyślnie false)



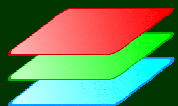
Zapis

- Zapis tekstu

```
plik.Write(tekst);
```

- Zapis tekstu i znaku końca linii

```
plik.WriteLine(tekst);
```



Odczyt

- Odczyt wiersza

```
var s = plik.ReadLine();
```

- Odczyt zadanej liczby znaków

```
var s = plik.Read(ile_znakow);
```

- Odczyt całości

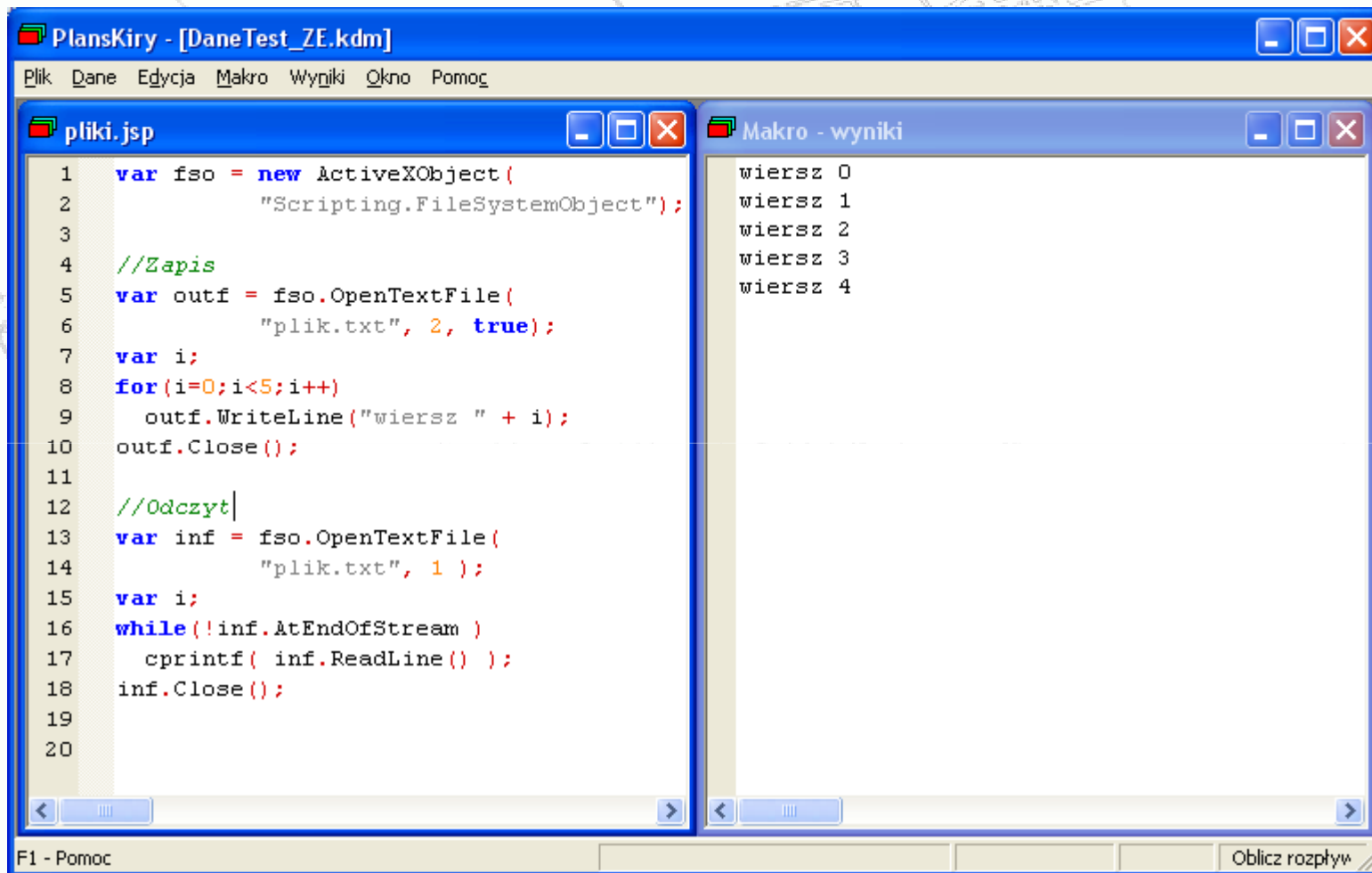
```
var s = plik.ReadAll();
```

- Test końca pliku

```
plik.AtEndOfStream
```



Zapis i odczyt

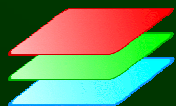


The screenshot shows the PlansKiry application window titled "PlansKiry - [DaneTest_ZE.kdm]". It has a menu bar with "Plik", "Dane", "Edycja", "Makro", "Wyniki", "Okno", and "Pomoc". The main area is split into two panes. The left pane, titled "pliki.jsp", contains a JavaScript script. The right pane, titled "Makro - wyniki", shows the output of the script. The script defines an ActiveXObject for file system operations, writes five lines to "plik.txt", and then reads them back. The output pane shows the five lines read.

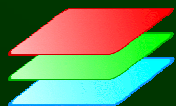
```
1  var fso = new ActiveXObject(  
2      "Scripting.FileSystemObject");  
3  
4  //Zapis  
5  var outf = fso.OpenTextFile(  
6      "plik.txt", 2, true);  
7  var i;  
8  for(i=0;i<5;i++)  
9      outf.WriteLine("wiersz " + i);  
10 outf.Close();  
11  
12 //Odczyt  
13 var inf = fso.OpenTextFile(  
14     "plik.txt", 1);  
15 var i;  
16 while(!inf.AtEndOfStream )  
17     cprintf( inf.ReadLine() );  
18 inf.Close();  
19  
20
```

wiersz 0
wiersz 1
wiersz 2
wiersz 3
wiersz 4

F1 - Pomoc Oblicz rozpiętość



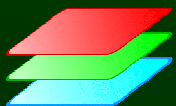
OBIEKTY (STRUKTURY)



Struktury

- Złożony typ danych, np.:
- Grupuje dane różnego typu, ale logicznie powiązane
- Składa się z **pól**
- Pola definiuje użytkownik
- Przykład:

Generator
Sn,
Pmax,
Pmin,
...



Obiekty w JScript

- Tworzenie obiektu:

```
var gen = new Object();
```

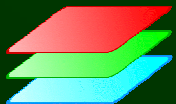
- Dodawanie pól:

```
gen.Sn = 250;
```

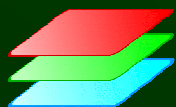
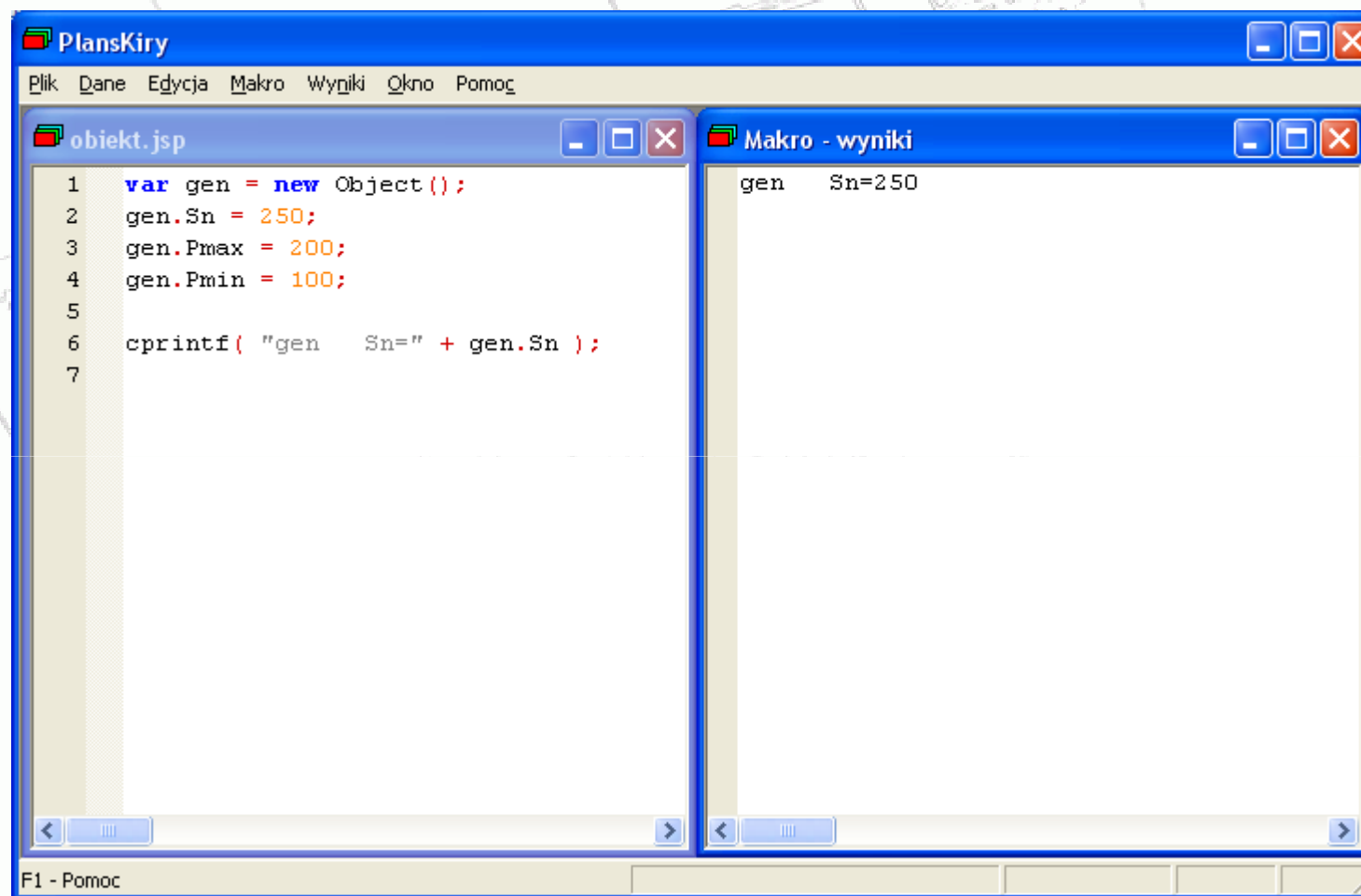
```
gen.Pmax = 200;
```

- Odczyt wartości:

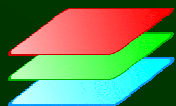
```
cprintf("Sn = " + Sn );
```



Obiekty

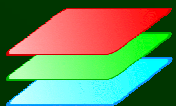


TABLICE



Tworzenie tablic

- Tworzenie pustej tablicy:
`var tab1 = new Array();`
- Tworzenie tablicy o zadanym rozmiarze:
`var tab2 = new Array(rozmiar);`
- Tworzenie wypełnionej tablicy
`var tab3 = new Array(el1,el2,el3 ...);`



Wstawianie wartości

- Dodanie wartości na koniec
`tab.push(wartosc)`
- Zmiana wartości
`tab[indeks]=wartosc;`
(jeżeli `indeks < rozmiar`, to rozmiar tablicy zostanie automatycznie zwiększony)
- Łączenie tablic
`tab.concat(tab1, tab2, tab3);`



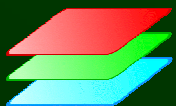
Dostęp do wartości

- Rozmiar tablicy:

```
tab.length
```

- Dostęp do wartości:

```
var z = tab[indeks];
```



Przydatne funkcje

- Sortowanie

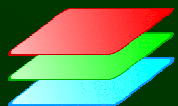
```
tab.sort();
```

```
tab.sort( funkcja_sortująca );
```

- Konwersja na tekst

```
tab.join( separator );
```

Separatorem może być znak końca linii: `"\n"`



Tablica struktur (obiektów)

- Dodanie do tablicy obiektu

```
var tab = new Array();
```

```
var gen1 = new Object();
```

```
gen1.Sn = 250;
```

```
gen1.Pmax = 200;
```

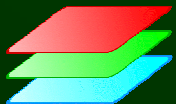
```
tab.push(gen1);
```

```
var gen2 = new Object();
```

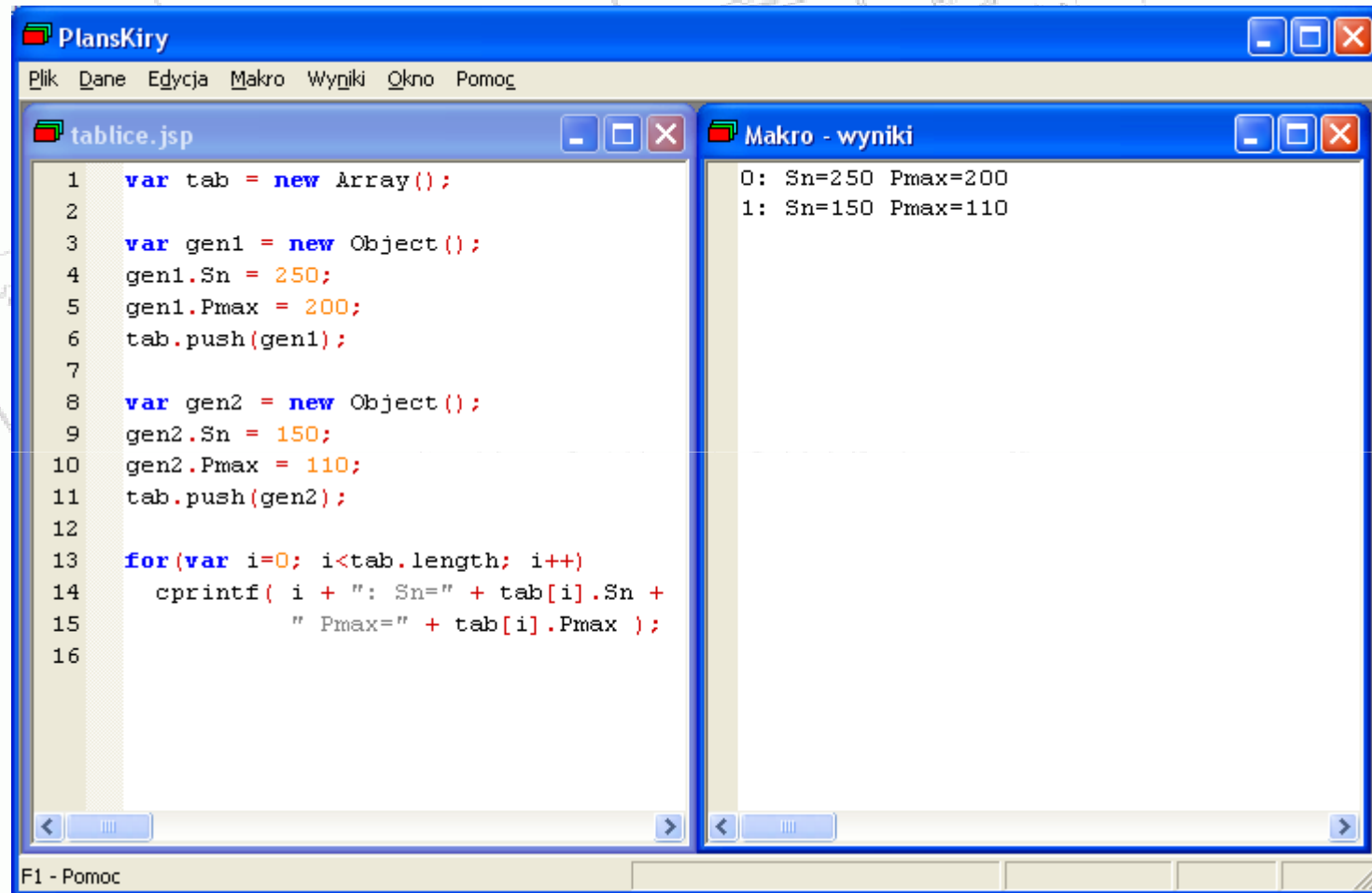
```
gen2.Sn = 150;
```

```
gen2.Pmax = 110;
```

```
tab.push(gen2);
```



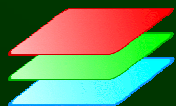
Tablica struktur (obiektów)




The screenshot shows the PlansKiry application interface. The main window has a menu bar with 'Plik', 'Dane', 'Edycja', 'Makro', 'Wyniki', 'Okno', and 'Pomoc'. Below the menu bar, there are two panes. The left pane, titled 'tablice.jsp', contains a JavaScript script. The right pane, titled 'Makro - wyniki', displays the output of the script. The script defines an array 'tab' and two objects 'gen1' and 'gen2', pushes them into the array, and then iterates over the array to print the values of 'Sn' and 'Pmax' for each object. The output shows two lines: '0: Sn=250 Pmax=200' and '1: Sn=150 Pmax=110'.

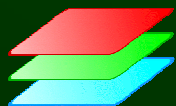
```
1 var tab = new Array();
2
3 var gen1 = new Object();
4 gen1.Sn = 250;
5 gen1.Pmax = 200;
6 tab.push(gen1);
7
8 var gen2 = new Object();
9 gen2.Sn = 150;
10 gen2.Pmax = 110;
11 tab.push(gen2);
12
13 for(var i=0; i<tab.length; i++)
14     cprintf( i + ": Sn=" + tab[i].Sn +
15             " Pmax=" + tab[i].Pmax );
16
```

0: Sn=250 Pmax=200
1: Sn=150 Pmax=110

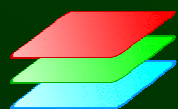




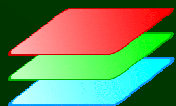
FUNKCJE UDOSTĘPNIONE PRZEZ PROGRAM PLANS



Nazwa	Opis
AddDataKDM	Doczytanie sieci do aktualnego modelu
CalcLF	Obliczenie rozptywu mocy
cprintf	Wypisanie komunikatu na ekranie
GetFile	Odszukanie pliku na dysku
ReadDataBIN	Wczytanie danych z pliku binarnego
ReadDataEPC	Wczytanie danych z pliku EPC
ReadDataKDM	Wczytanie danych z pliku KDM
ReadDataIEN	Wczytanie danych z pliku IEN
ReadDataUZP	Doczytanie uzupełnień
ReadDataUCTE	Wczytanie danych z pliku UCTE
SaveDataBIN	Zapisanie modelu do pliku binarnego
SaveDataEPC	Zapisanie modelu do pliku EPC
SaveDataIEN	Zapisanie modelu do pliku IEN
SaveDataKDM	Zapisanie modelu do pliku KDM
SaveDataUCTE	Zapisanie modelu do pliku UCTE
TestConnection	Test spójności
TestRX	Test R/X



DOSTĘP DO DANYCH SIECIOWYCH

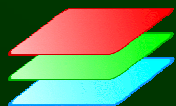


Języki skryptowe w programie Plans

Tomasz Zdun

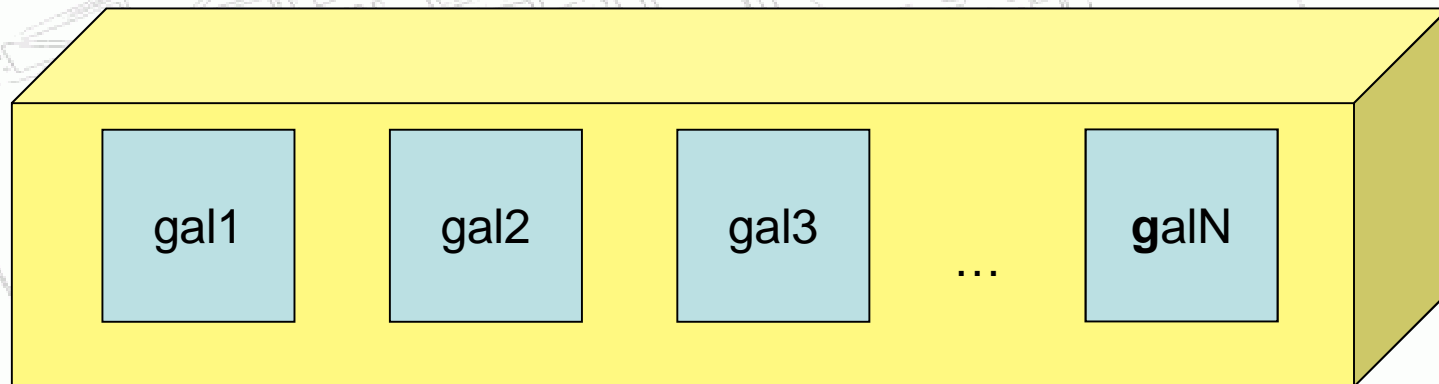
Udostępnione tablice i struktury

Nazwa	Opis
BraArray	Tablica gałęzi
Calc	Struktura zawierająca parametry procesu obliczeniowego
Data	Struktura zawierająca informacje o modelu
GenArray	Tablica generatorów
LinArray	Tablica linii
LodArray	Tablica odbiorów
Main	Struktura zawierająca informacje o programie
NodArray	Tablica węzłów
Rslt	Struktura zawierająca informacje o wyniku procesu obliczania rozptywu mocy
TrfArray	Tablica transformatorów
ZonArray	Tablica obszarów

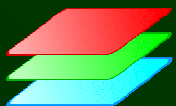


Tablica gałęzi

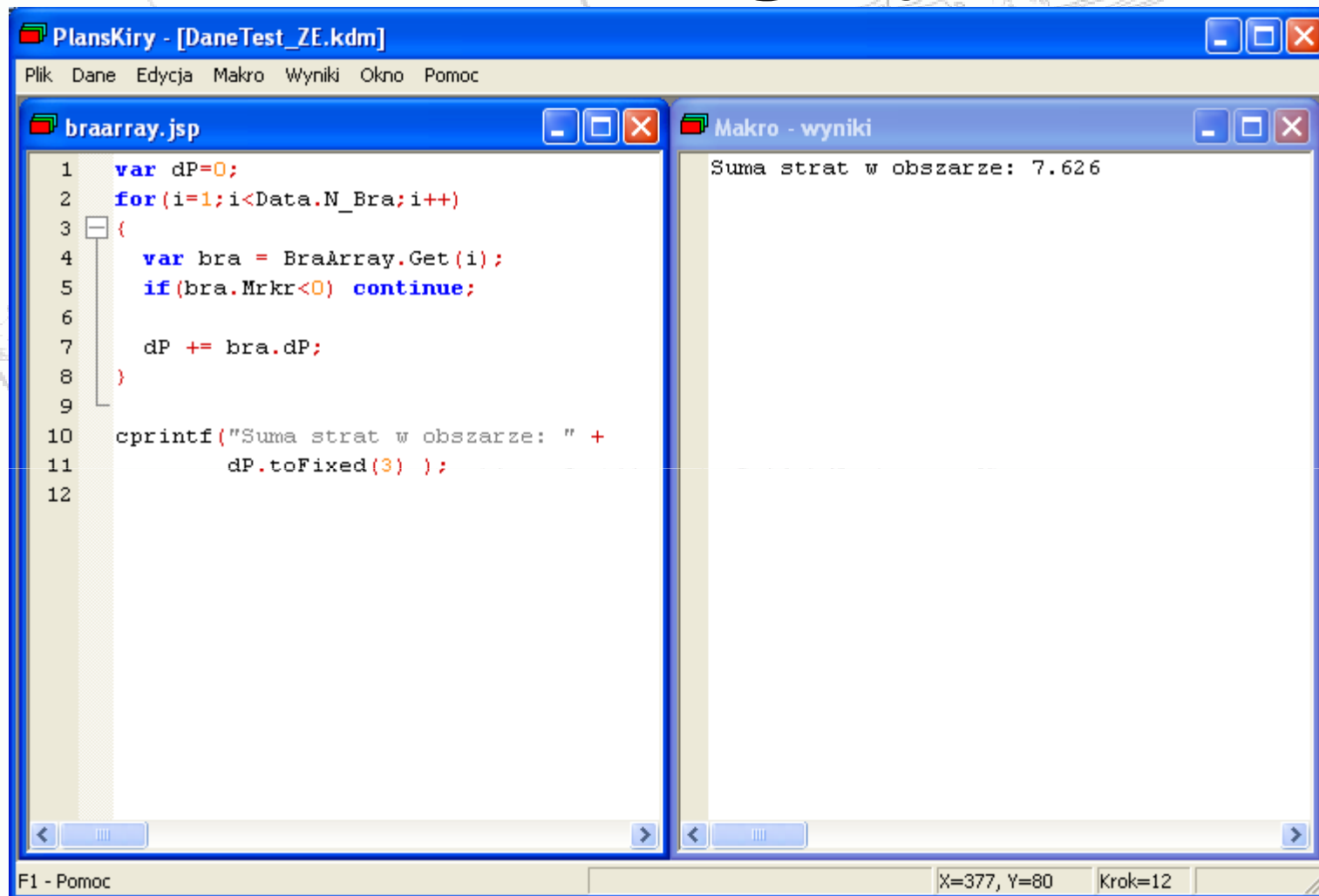
- Tablica gałęzi BraArray zawiera wszystkie gałęzie.



- Dostęp do gałęzi poprzez metody Get i Find



Tablica gałęzi



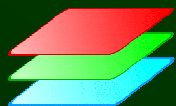
The screenshot displays the PlansKiry IDE interface. The main editor window, titled 'braarray.jsp', contains the following JavaScript code:

```
1  var dP=0;
2  for (i=1;i<Data.N_Bra;i++)
3  {
4      var bra = BraArray.Get(i);
5      if (bra.Mrkr<0) continue;
6
7      dP += bra.dP;
8  }
9
10 cprintf("Suma strat w obszarze: " +
11         dP.toFixed(3) );
12
```

To the right, a window titled 'Makro - wyniki' shows the output of the macro execution:

```
Suma strat w obszarze: 7.626
```

The status bar at the bottom indicates 'F1 - Pomoc', 'X=377, Y=80', and 'Krok=12'.



Jedna gałąź

```
var galaz1 = BraArray.Get(indeks);  
var galaz2 = BraArray.Find("L10");
```

- **Nazwa:**

```
cprintf(galaz1.Name);
```

- **Reaktancja:**

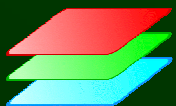
```
cprintf(galaz1.Xb.tofixed(1));
```

- **Status:**

```
cprintf(galaz1.St);
```

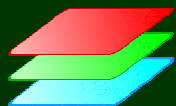
- **Wyłączenie gałęzi:**

```
galaz1.St=-1;
```





PLANS JAKO SERWER AUTOMATYZACJI



Języki skryptowe w programie Plans

Tomasz Zdun

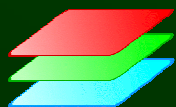
Uruchomienie drugiego Plansa

- Uruchomienie:

```
var plans2 = new ActiveXObject(  
    "PlanSEMC.Application");
```

- Praca:

```
plans2.ReadDataKDM( " " );  
plans2.CalcLF( " " );
```



Excel w Plansie

- Uruchomienie:

```
var ex = new ActiveXObject(  
    "Excel.Application");  
ex.visible = true; //excel widoczny
```

- Utworzenie nowego arkusza

```
ex.Workbooks.Add();
```

- Dostęp do komórek w arkuszu

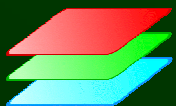
```
ex.Cells(1, 1).Value = 123.4;
```

- Zapis arkusza i zamknięcie Excela

```
ex.Save();
```

```
ex.Quit();
```

- Więcej w dokumentacji Excela



Plans z zewnątrz

- Uruchomienie ze skryptu zewnętrznego (plik *plans2_z_zewnatrz.js*):

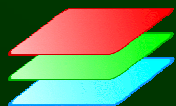
```
var plans = new ActiveXObject(  
    "PlansEMC.Application");
```

- Praca z Plansem:

```
plans.ReadDataKDM( " " );  
plans.CalcLF();  
plans.BraArray.Find("C006, ").St=-1;  
plans.CalcLF();  
...
```

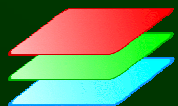



DOKUMENTACJA



Dokumentacja

- Funkcje programu Plans:
plik ***MakraJS.chm***
- Strony WWW
 - <http://msdn.microsoft.com>
[http://msdn.microsoft.com/en-us/library/ff729665\(v=VS.94\).aspx](http://msdn.microsoft.com/en-us/library/ff729665(v=VS.94).aspx)
 - <http://www.poradnik-webmastera.com/kursy/javascript/>





PRZYKŁADY WYKORZYSTANIA SKRYPTÓW

