

Język skryptowy Plezuro
Dokumentacja

P l e z u r o

1 Autorzy

Uczelnia : Politechnika Śląska
Wydział : Wydział Matematyki Stosowanej
Rok akademicki : 2013/2014
Kierunek : Informatyka
Semestr : IV
Nazwiska

- Piotr Sroczkowski
Pomysł, język skryptowy, IDE, dokumentacja, prawie wszystko
- Daniel Mikulski
Niektóre skrypty

2 Dane techniczne

Język : c# 5.0
Platforma : Mono 3.2.8
Kompilator : gmcs 3.2.8.0
System kontroli wersji : git 1.9.1
Adres publicznego repozytorium : <https://github.com/oprogramador/repo>
Licencja : GNU GPL 2.0

3 Dla użytkownika

3.1 Krótki opis

Został zaimplementowany język skryptowy. W oparciu na nim działa nierelacyjna baza danych.

3.2 Krótki przewodnik

3.2.1 Prosty przykład

```
$i=2;
$n=0;
{n<first }.while({
    $k := 2;
    $ispr = true;
    {k*k<=i }.while({
        {i%k==0}.if({ ispr=false });
        k++
    });
    { ispr }.if({ n++});
    i++
});
i-1
```

3.2.2 Komentarze

```
//this is a comment

/*
Another comment
*/
```

3.2.3 Zmienne

Przy deklaracji zmiennej, piszemy znak '\$', określa to zasięg zmiennej.

```
$a = 12;
a++;
$b = a*2;
a+b^3
```

3.2.4 Klonowanie a referencja

```
$a = 21;
$b = a;
b++;
b.::println(); //it prints '22'
(a==b).::println(); //it prints 'true'

$c = 4;
$d := c;
d++;
c.::println(); //it prints '4'
(c==d).::println(); //it prints 'false'

$e = 2;
$f = e.::clone();
f++;
e.::println(); //it prints '2'
(e==f).::println(); //it prints 'false'

(1==1).::println(); //it prints 'true'
(1==1).::println(); //it prints 'false'
```

null

3.2.5 Wbudowane klasy (typy)

```
//number
$x = 2.3e45;
$y = 0xff; //hexadecimal
$z = 072; //octal
$a = 0b11011; //binary

//string
$b = 'aaaaaaaaaaaaaaaaaaaa';
$c = "wfefwfwf";
$cc = '''xxx
yyy
zzz''';

//list
$d = [1,2,3,4];

//dictionary
$f = #[1,2,3,4];
```

```

//set
$g = $[3,4,5];

//error
$h = 1/0;

//class
$i = 1::class();

//package
$j = i::package();

//pair
$k = 3:4;

//procedure
$l = {1+2};

(x,y,z,a,b,c,cc,d,e,f,g,h,i,j,k,l)

```

3.2.6 Indeksowanie

```

'abcdefghijklmnpqr'[( [1..5,0,0,2..12..3])].toS.printl;
[12,13,14,15,16][1..4]

```

3.2.7 Krotki

```

($a, $b) = (1,3);
a::printl();
b::printl();

$c = 5;
$d = 6;
(a,b,c,d) = (b,c,d,a);

a::printl();
(a,b,c,d)::printl();

a <-> c;
(a,b,c,d)

```

3.2.8 Wyrażenia warunkowe

```

$x = 90;
$y = true;
{x>0}.if{
    'aa'.printl
}.if({x<40},{
    'bb'.printl
}).elif({y},{
    'cc'.printl
}).else{
    'dd'.printl
}

```

3.2.9 Pętle

```
$i = 13;
{i<20}.while({
    i.println;
    i++;
});

[1,2,3,4,5].each({ this.println });

[108,2,3,4,5,20,90].each({ args.println });

30..70..6.5.each({ args.println })
```

3.2.10 Procedury

```
$f = {
    ($x,$y,$z) = args;
    ::println ( 'args='+args );
    ::println ( 'x='+x );
    ::println ( 'y='+y );
    ::println ( 'z='+z );
    x+y*z};

f:::applyF([2,3,4]):::println();
f(4,5,6):::println();
f:::time():::println(); //executing time in milliseconds; x,y,z are undefined here
{f(4,5,6)}:::time():::println(); //executed time; x,y,z are defined
```

3.2.11 Operacje na plikach

```
$txt = 'abc.txt'.fromF;
{txt.class==String}.if({ (txt*4).toF('xyz.txt') })
```

3.2.12 Generowanie tabelki html

```
[[[1,2,3]],[4,5]].html.toF('1.html');
[:::dic('name','Jewwwwan','city','Marseille'), :::dic('name','Tom','city','Miami','sex'
```

3.2.13 Klasy zdefiniowane przez użytkownika

```
Lang
<< 'Person':::newClass($[Object],#[
    'init',{
        @this << ('age': (vals*2))
    },
    '+',{
        (@this['age'] += vals)
    },
    "get-age",{
        @this['age']
    },
    "set-age",{
        @this['age'] = vals
    },
    "str",{
```

```

        "I'm"+(@this[ 'age' ])+ ' years old.'
    },
    "destroy",{
        :: printl( 'person_destroy' );
    }
]);
$Per = $[ @Lang[ 'Person' ] ];
Lang << 'Dog' :: newClass( Per, #[
    'init',{
        ($age, $race) = vals;
        @(@Lang[ 'Person' ])[ 'init' ]( this, age );
        @this << ( 'race': race );
    }
]);

$p = (@Lang[ 'Person' ](14));
@p[ 'age' ]++;
p+50;
$d = (@Lang[ 'Dog' ](13, 'Akbash' ));
d+3;
( ' '+d ) :: printl ();
//((@Lang[ 'Person' ]) :: set( 'age' ))(d,100);
@p,@d

```

3.2.14 Kolejność operatorów dwuargumentowych (od tych wykonywanych na końcu)

```

;
:=
=
,
<->
<<
>>
?
|
&
<=>
>=
>
<=
<
!=
==
===
=~
+
-
%

```

*
 /
 ^
 Razem
 • ^^
 • =~
 ..
 :

3.2.15 Operatory jednoargumentowe

!
 &&
 **
 #
 ++
 --
 @

3.2.16 Wbudowane pakiety, klasy, metody, operatory i stałe

- package Lang
 - class Boolean
 - Dziedziczy po: [Object]
 - * Krótki opis : Wartość logiczna
 - * Operatory:
 - ?
 - Argumenty: (Boolean b, Pair p)
 - Typ zwracany: Object
 - Krótki opis : Gdy b ma wartość *true* zwraca pierwszą wartość z pary p, w przeciwnym razie zwraca drugą wartość.
 - |
 - Argumenty: (Boolean a, Boolean b)
 - Typ zwracany: Boolean
 - Krótki opis : Alternatywa logiczna
 - &
 - Argumenty: (Boolean a, Boolean b)
 - Typ zwracany: Boolean
 - Krótki opis : Koniunkcja logiczna
 - !
 - Argumenty: (Boolean b)
 - Typ zwracany: Boolean
 - Krótki opis : Przeczenie logiczne
 - * Metody:
 - if
 - Argumenty: (Boolean b, Procedure t, Procedure f)
 - Typ zwracany: Object
 - Krótki opis : Instrukcja warunkowa - jeśli b ma wartość *true*, wykonywana jest procedura t, w przeciwnym razie wykonywana jest procedura f.

- * Stałe:
 - true
Krótki opis : Prawda
 - false
Krótki opis : Fałsz
- class Class

Dziedziczy po: [Object]

 - * Krótki opis : Klasa
 - * Metody:
 - parents
Argumenty: (Class c)
Typ zwracany: List
Krótki opis : Zwraca wszystkie klasy bazowe (występuje dziedziczenie po wielu klasach).
 - package
Argumenty: (Class c)
Typ zwracany: Package
Krótki opis : Zwraca pakiet, do którego należy klasa.
- class Dictionary

Dziedziczy po: [Object]

 - * Krótki opis : Kolekcja słownik
 - * Operatory:
 - <<
Argumenty: (Dictionary d, Pair p)
Typ zwracany: Dictionary
Krótki opis : Dodaje parę klucz-wartość do słownika.
 - * Metody:
 - ref
Argumenty: (Dictionary d, Object key)
Typ zwracany: Object
Krótki opis : Zwraca referencję do wartości zadanej przez klucz.
 - len
Argumenty: (Dictionary d)
Typ zwracany: Number
Krótki opis : Zwraca długość słownika.
 - contains
Argumenty: (Dictionary d, Object key)
Typ zwracany: Boolean
Krótki opis : Informacja, czy słownik zawiera podany klucz
 - keys
Argumenty: (Dictionary d)
Typ zwracany: List
Krótki opis : Zwraca listę wszystkich kluczy.
 - remove
Argumenty: (Dictionary d, Object key)
Typ zwracany: Dictionary
Krótki opis : Zwraca nowy słownik z usuniętym kluczem.
- class DotFunc

Dziedziczy po: [Object]

 - * Krótki opis : Para (funkcja, pierwszy argument)
 - * Operatory:

- `..`
 - Argumenty: (DotFunc d, Object o)
 - Typ zwracany: Object
 - Krótki opis : Wywołuje funkcję dla podanych argumentów. Pierwszy argument jest zapamiętany, kolejne zawarte są w obiekcie o (obiekt klasy Empty traktowany jest jako brak argumentu, Tuple jako wiele argumentów, zaś innych klas jako pojedynczy argument).
- class Empty
 - Dziedziczy po: [Object]
 - * Krótki opis : Pusta wartość
 - * Metody:
 - array
 - Argumenty: (Empty e)
 - Typ zwracany: List
 - Krótki opis : Zwraca pustą listę.
- class Error
 - Dziedziczy po: [Object]
 - * Krótki opis : Błąd
 - * Metody:
 - msg
 - Argumenty: (Error e)
 - Typ zwracany: String
 - Krótki opis : Zwraca komunikat błędu.
- class List
 - Dziedziczy po: [Object]
 - * Krótki opis : Kolekcja lista
 - * Operatory:
 - `<<`
 - Argumenty: (List l, Object o)
 - Typ zwracany: List
 - Krótki opis : Akcja *push* - wrzucenie obiektu o do listy l.
 - `>>`
 - Argumenty: (List l, Reference r)
 - Typ zwracany: List
 - Krótki opis : Akcja *pop* - zrzucenie obiektu z listy l do referencji r.
 - `+`
 - Argumenty: (List a, List b)
 - Typ zwracany: List
 - Krótki opis : Konkatenacja dwóch list.
 - `*`
 - Argumenty: (List l, Number n)
 - Typ zwracany: Object
 - Krótki opis : n-krotne kopiowanie tablicy l.
 - * Metody:
 - get
 - Argumenty: (List l, Number n)
 - Typ zwracany: Object
 - Krótki opis : Zwraca n-ty element listy l.
 - len
 - Argumenty: (List l)

- Typ zwracany: Number
 Krótki opis : Zwraca długość tablicy l.
- ref
 Argumenty: (List l, Number n)
 Typ zwracany: Reference
 Krótki opis : Zwraca referencję do n-tego elementu listy l.
 - each
 Argumenty: (List l, Procedure p)
 Typ zwracany: Object
 Krótki opis : Iteracja listy l, wykonywanie procedury p dla każdego elementu.
 - where
 Argumenty: (List l, Procedure p)
 Typ zwracany: List
 Krótki opis : Selekcja elementów, dla których procedura p zwraca wartość *true*.
 - map
 Argumenty: (List l, Procedure p)
 Typ zwracany: List
 Krótki opis : Mapowanie procedury p po liście l.
 - sort
 Argumenty: (List l)
 Typ zwracany: List
 Krótki opis : Sortowanie.
 - orderBy
 Argumenty: (List l, Procedure p)
 Typ zwracany: List
 Krótki opis : Sortowanie według wartości zwracanej przez procedurę p.
 - orderByD
 Argumenty: (List l, Procedure p)
 Typ zwracany: List
 Krótki opis : To samo co orderBy, ale w odwrotnej kolejności.
 - groupBy
 Argumenty: (List l, Procedure p)
 Typ zwracany: List
 Krótki opis : Grupowanie według wartości zwracanej przez procedurę p.
 - reverse
 Argumenty: (List l)
 Typ zwracany: List
 Krótki opis : Odwracanie listy.
 - max
 Argumenty: (List l)
 Typ zwracany: Object
 Krótki opis : Zwraca maksymalną wartość.
 - min
 Argumenty: (List l)
 Typ zwracany: Object
 Krótki opis : Zwraca minimalną wartość.
 - median
 Argumenty: (List l)
 Typ zwracany: Object
 Krótki opis : Zwraca medianę.
 - remove
 Argumenty: (List l, Number n)

- Typ zwracany: List
 - Krótki opis : Zwraca listę z usuniętym elementem w indeksie n.
- toSet
 - Argumenty: (List l)
 - Typ zwracany: Set
 - Krótki opis : Konwertuje do zbioru (*set*).
- html
 - Argumenty: (List l)
 - Typ zwracany: String
 - Krótki opis : Zwraca tabelką w html.
- class NullClass
 - Dziedziczy po: [Object]
 - * Krótki opis : Wartość null
 - * Stałe:
 - null
 - Krótki opis : Null
- class Number
 - Dziedziczy po: [Object]
 - * Krótki opis : Liczba rzeczywista
 - * Operatory:
 - +
 - Argumenty: (Number a, Number b)
 - Typ zwracany: Number
 - Krótki opis : Dodawanie.
 - -
 - Argumenty: (Number a, Number b)
 - Typ zwracany: Number
 - Krótki opis : Odejmowanie.
 - *
 - Argumenty: (Number a, Number b)
 - Typ zwracany: Number
 - Krótki opis : Mnożenie.
 - /
 - Argumenty: (Number a, Number b)
 - Typ zwracany: Number
 - Krótki opis : Dzielenie.
 - ^
 - Argumenty: (Number a, Number b)
 - Typ zwracany: Number
 - Krótki opis : Potęgowanie.
 - ++
 - Argumenty: (Number a)
 - Typ zwracany: Number
 - Krótki opis : Inkrementacja.
 - --
 - Argumenty: (Number a)
 - Typ zwracany: Number
 - Krótki opis : Dekrementacja.
 - * Metody:
 - chr
 - Argumenty: (Number n)

- Typ zwracany: String
 Krótki opis : Zwraca znak o podanym kodzie ASCII n.
- sin
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Sinus.
 - cos
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Cosinus.
 - tan
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Tangens.
 - asin
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Arcus sinus.
 - acos
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Arcus cosinus.
 - atan
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Arcus tangens.
 - sinh
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Sinus hiperboliczny.
 - cosh
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Cosinus hiperboliczny.
 - tanh
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Tangens hiperboliczny.
 - round
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Zaokrąglenie.
 - floor
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Zaokrąglenie w dół.
 - ceil
 Argumenty: (Number n)
 Typ zwracany: Number
 Krótki opis : Zaokrąglenie do góry.
 - abs
 Argumenty: (Number n)

- Typ zwracany: Number
 - Krótki opis : Wartość absolutna.
- ln
 - Argumenty: (Number n)
 - Typ zwracany: Number
 - Krótki opis : Logarytm naturalny.
- sqrt
 - Argumenty: (Number n)
 - Typ zwracany: Number
 - Krótki opis : Pierwiastek kwadratowy.
- fib
 - Argumenty: (Number n)
 - Typ zwracany: Number
 - Krótki opis : N-ty element ciągu Fibonacciego.
- * Stałe:
 - pi
 - Krótki opis : Liczba pi
 - e
 - Krótki opis : Liczba e
- class Object
 - Dziedziczy po: []
 - * Krótki opis : Dowolny obiekt
 - * Operatory:
 - .
 - Argumenty: (Object a, SoftLink s)
 - Typ zwracany: DotFunc
 - Krótki opis : Tworzenie obiektu DotFunc.
 - ;
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Object
 - Krótki opis : Zwraca obiekt b.
 - ,
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Object
 - Krótki opis : Tworzenie krotki.
 - ;
 - Argumenty: (Reference a, Reference b, Reference c)
 - Typ zwracany: Number
 - Krótki opis : Zamiana miejscami zmiennych a oraz b.
 - :
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Pair
 - Krótki opis : Tworzenie pary.
 - <=>
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Number
 - Krótki opis : Zwraca 1 gdy a jest większe od b, 0 gdy równe, -1 gdy mniejsze.
 - >=
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy a jest większe bądź równe b.

- >
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy a jest większe od b.
- <=
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy a jest mniejsze bądź równe b.
- <
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy a jest mniejsze od b.
- !=
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy a jest różne od b.
- ==
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy a równe b.
- ===
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy a jest b (ten sam obiekt).
- &&
 - Argumenty: (Reference r)
 - Typ zwracany: Pointer
 - Krótki opis : Zwraca wskaźnik do r.
- :=
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Object
 - Krótki opis : Klonowanie b do a, można klonować całe krotki.
- =
 - Argumenty: (Object a, Object b)
 - Typ zwracany: Boolean
 - Krótki opis : Przypisywanie b do a (referencja, można przypisywać całe krotki).
- * Metody:
 - class
 - Argumenty: (Object o)
 - Typ zwracany: Class
 - Krótki opis : Zwraca klasę obiektu o.
 - print
 - Argumenty: (Object o)
 - Typ zwracany: Object
 - Krótki opis : Wypisanie o do konsoli.
 - printl
 - Argumenty: (Object o)
 - Typ zwracany: Object
 - Krótki opis : Wypisanie o do konsoli jako nowej linii.
 - clone
 - Argumenty: (Object o)
 - Typ zwracany: Object
 - Krótki opis : Klonowanie.

- `lent`
Argumenty: (Object o)
Typ zwracany: Number
Krótki opis : Zwraca długość o (dla krotek (Tuple) długość krotki, dla obiektu Empty 0, dla obiektów innych klas 1).
- `set`
Argumenty: (Object o)
Typ zwracany: Set
Krótki opis : Tworzenie zbioru (*Set*).
- `dic`
Argumenty: (Object o)
Typ zwracany: Dictionary
Krótki opis : Tworzenie słownika.
- class `Package`
Dziedziczy po: [Object]
 - * Krótki opis : Pakiet (kolekcja klas i innych pakietów)
 - * Operatory:
 - * Metody:
 - `package`
Argumenty: (Package p)
Typ zwracany: Package
Krótki opis : Zwraca nadrzędny pakiet.
 - * Stałe:
 - `true`
Krótki opis : Prawda
 - `false`
Krótki opis : Fałsz
- class `Pair`
Dziedziczy po: [Object]
 - * Krótki opis : Uporządkowana para (klucz, wartość)
 - * Metody:
 - `key`
Argumenty: (Pair p)
Typ zwracany: Object
Krótki opis : Zwraca klucz.
 - `value`
Argumenty: (Pair p)
Typ zwracany: Object
Krótki opis : Zwraca wartość.
- class `Pointer`
Dziedziczy po: [Object]
 - * Krótki opis : Wskaźnik do obiektu
 - * Operatory:
 - `**`
Argumenty: (Pointer p)
Typ zwracany: Object
Krótki opis : Zwraca obiekt, na który wskazuje wskaźnik p.
- class `Procedure`
Dziedziczy po: [Object]
 - * Krótki opis : Procedura, która przyjmuje parametry i zwraca wartość

- * Metody:
 - apply
 - Argumenty: (Procedure p)
 - Typ zwracany: Object
 - Krótki opis : Wywołanie procedury bez parametrów.
 - applyF
 - Argumenty: (Procedure p, List l)
 - Typ zwracany: Object
 - Krótki opis : Wywołanie procedury z parametrów.
 - while
 - Argumenty: (Procedure a, Procedure b)
 - Typ zwracany: Object
 - Krótki opis : Pętla *while*, warunek określa procedura a, w pętli wykonywana jest procedura b.
 - integral
 - Argumenty: (Procedure p, Number beg, Number end)
 - Typ zwracany: Object
 - Krótki opis : Całkowanie numeryczne.
 - time
 - Argumenty: (Procedure p)
 - Typ zwracany: Number
 - Krótki opis : Zlicza czas wykonywania procedury p w milisekundach.
- class Reference
 - Dziedziczy po: [Object]
 - * Krótki opis : Referencja do obiektu, pomocnicza klasa, każdy obiekt ma referencję, ale żaden obiekt nie jest klasy Reference.
- class Set
 - Dziedziczy po: [Object]
 - * Krótki opis : Kolekcja zbiorów
 - * Operatory:
 - <<
 - Argumenty: (Set s, Object o)
 - Typ zwracany: Object
 - Krótki opis : Wrzucenie obiektu o do zbioru s
 - * Metody:
 - len
 - Argumenty: (Set s)
 - Typ zwracany: Object
 - Krótki opis : Zwraca długość zbioru.
 - max
 - Argumenty: (Set s)
 - Typ zwracany: Object
 - Krótki opis : Zwraca maksymalną wartość.
 - min
 - Argumenty: (Set s)
 - Typ zwracany: Object
 - Krótki opis : Zwraca minimalną wartość.
 - contains
 - Argumenty: (Set s, Object o)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy zbiór zawiera podaną wartość.

- join
 - Argumenty: (Set a, Set b)
 - Typ zwracany: Set
 - Krótki opis : Iloczyn zbiorów.
- except
 - Argumenty: (Set a, Set b)
 - Typ zwracany: Set
 - Krótki opis : Różnica zbiorów.
- union
 - Argumenty: (Set a, Set b)
 - Typ zwracany: Set
 - Krótki opis : Suma zbiorów.
- remove
 - Argumenty: (Set s, Object o)
 - Typ zwracany: Object
 - Krótki opis : Zwraca zbiór z usuniętą wartością.
- toList
 - Argumenty: (Set s)
 - Typ zwracany: Object
 - Krótki opis : Konwersja do listy.
- len
 - Argumenty: (Set s)
 - Typ zwracany: Object
 - Krótki opis : Zwraca długość zbioru.
- class SoftLink
 - Dziedziczy po: [Object]
 - * Krótki opis : Link symboliczny
 - * Operatory:
 - ^^
 - Argumenty: (SoftLink s, Object o)
 - Typ zwracany: Object
 - Krótki opis : Wykonanie procedury wskazywanej przez link dla podanych argumentów.
- class String
 - Dziedziczy po: [Object]
 - * Krótki opis : Łańcuch tekstowy
 - * Operatory:
 - +
 - Argumenty: (String s, Object o)
 - Typ zwracany: String
 - Krótki opis : Konkatenacja.
 - *
 - Argumenty: (String s, Number n)
 - Typ zwracany: String
 - Krótki opis : N-krotne kopiowanie.
 - #
 - Argumenty: (String s)
 - Typ zwracany: Object
 - Krótki opis : Podstawienie obliczonych wartości w środku stringa.
 - =~
 - Argumenty: (String regex, String s)
 - Typ zwracany: Boolean
 - Krótki opis : Informuje czy string s zawiera wyrażenie regularne regex.

- * Metody:
 - len
 - Argumenty: (String s)
 - Typ zwracany: Number
 - Krótki opis : Zwraca długość stringa.
 - get
 - Argumenty: (String s, Number n)
 - Typ zwracany: String
 - Krótki opis : Zwraca n-ty znak.
 - reverse
 - Argumenty: (String s)
 - Typ zwracany: String
 - Krótki opis : Zwraca odwróconego stringa.
 - ord
 - Argumenty: (String s)
 - Typ zwracany: Number
 - Krótki opis : Zwraca kod ASCII pierwszego znaku.
 - fromF
 - Argumenty: (String s)
 - Typ zwracany: String
 - Krótki opis : Wczytuje zawartość pliku do stringa.
 - toF
 - Argumenty: (String s, String f)
 - Typ zwracany: Boolean
 - Krótki opis : Zapisuje stringa s do pliku f, zwracana wartość informuje czy zapis się udał.
 - put
 - Argumenty: (String f, String s)
 - Typ zwracany: Boolean
 - Krótki opis : Zapisuje stringa s do pliku f, zwracana wartość informuje czy zapis się udał.
 - putA
 - Argumenty: (String f, String s)
 - Typ zwracany: Boolean
 - Krótki opis : Dopisuje stringa s do pliku f, zwracana wartość informuje czy zapis się udał.
 - append
 - Argumenty: (String s, String f)
 - Typ zwracany: Boolean
 - Krótki opis : Dopisuje stringa s do pliku f, zwracana wartość informuje czy zapis się udał.
 - load
 - Argumenty: (String s)
 - Typ zwracany: Object
 - Krótki opis : Wykonuje moduł zapisany w pliku.
 - eval
 - Argumenty: (String s)
 - Typ zwracany: Object
 - Krótki opis : Wykonuje kod zawarty w stringu.
- class Tuple
 - Dziedziczy po: [Object]
 - * Krótki opis : Kolekcja krotka, każda krotka posiada przynajmniej 2 elementy.

4 Dla programisty

4.1 Jak ściągnąć, skompilować i uruchomić?

1. Zainstaluj dowolną dystrybucję systemu operacyjnego GNU/Linux (dalsze instrukcje dla pochodnych Debiana). Możesz skorzystać ze strony : <http://www.linuxmint.com/download.php>.
2. Zainstaluj mono. Użyj polecenia terminala : *sudo apt-get install monodevelop mono-complete*.
3. Zainstaluj git : *sudo apt-get install git*
4. Utwórz nowy folder i wejdź do niego : *mkdir project1; cd project1*
5. Ściągnij projekt : *git download https://github.com/oprogramador/repo.git; cd repo*
6. Skompiluj : *./make.sh*
7. Uruchom : *./plezuro.exe*

Możesz również spróbować skompilować w systemie Windows używając Visual Studio lub Mono.

4.2 Kod

4.2.1 Pliki, przestrzenie nazw (odpowiadają folderom), klasy, interfejsy, enumeracje, dziedziczenie

```
lib/Co.cs: class Co
lib/HtmlArrayTable.cs: class HtmlArrayTable : HtmlTable
lib/ITypeConvertible.cs: interface ITypeConvertible
lib/HtmlTableFactory.cs: static class HtmlTableFactory
lib/MyCrypto.cs: class MyCrypto
lib/HtmlDicTable.cs: class HtmlDicTable : HtmlTable
lib/Integral.cs: class Integral
lib/SimpleTypeConverter.cs: static class SimpleTypeConverter
lib/HtmlTable.cs: abstract class HtmlTable
lib/SString.cs: class SString
Engine/StaticParser.cs: static class StaticParser
Engine/Parser.cs: class Parser
Engine/Evaluator.cs: class Evaluator : IPrintable
Engine/IOutputable.cs: interface IOutputable : ITexttable
Engine/RPN.cs: class RPN
Engine/Tokenizer.cs: class Tokenizer
Engine/ErrorMessage.cs: class ErrorMessage
Engine/Engine.cs: class Engine
Engine/SyntaxException.cs: class SyntaxException : Exception
Engine/IOMap.cs: class IOMap : Dictionary<ITexttable, IOutputable>, IRefreshable
Engine/SymbolMap.cs: class SymbolMap : ConcurrentDictionary<string, object>
Engine/SymbolException.cs: class SymbolException : Exception
Engine/RPNTypes.cs: enum RPNTypes
Engine/TokenConverter.cs: class TokenConverter
Program.cs: class Program
Tests/TestUnit.cs: class TestUnit
DataFixtures/DataFixtures.cs: class DataFixtures : SetT
MyTypes/MyClasses/ClassT.cs: public class ClassT : IItem, IVariable, ICallable
MyTypes/MyClasses/DictionaryT.cs: public class DictionaryT : SortedDictionary<IVariable, IVariable>, IVariable
MyTypes/MyClasses/PointerT.cs: class PointerT : Pointer<ReferenceT>, IVariable
MyTypes/MyClasses/DotFunc.cs: class DotFunc : IVariable, ICallable
```

```

MyTypes/MyClasses/ErrorT.cs: class ErrorT : IVariable
MyTypes/MyClasses/TupleT.cs: public class TupleT : SList<IVariable>, IVariable
MyTypes/MyClasses/NullType.cs: class NullType : IVariable
MyTypes/MyClasses/Number.cs: class Number : Pointer<double>, IVariable
MyTypes/MyClasses/StringT.cs: class StringT : Pointer<string>, IVariable, IIndexable
MyTypes/MyClasses/StringT.cs: class MiniParser : IParseable
MyTypes/MyClasses/ReferenceT.cs: public class ReferenceT : Pointer<IVariable>, IVariable, ITypeConversion
MyTypes/MyClasses/MyRandom.cs: class MyRandom : Number
MyTypes/MyClasses/CallFunc.cs: class CallFunc : IVariable
MyTypes/MyClasses/ProcedureT.cs: public class ProcedureT : OStack, ICallable
MyTypes/MyClasses/IfObject.cs: class IfObject : IVariable
MyTypes/MyClasses/StopPoint.cs: class StopPoint : IVariable
MyTypes/MyClasses/PairT.cs: public class PairT : IVariable
MyTypes/MyClasses/SoftLink.cs: class SoftLink : Pointer<string>, IVariable
MyTypes/MyClasses/MyObject.cs: class MyObject : IVariable
MyTypes/MyClasses/BooleanT.cs: class BooleanT : Pointer<bool>, IVariable
MyTypes/MyClasses/EmptyT.cs: class EmptyT : IVariable
MyTypes/MyClasses/RangeT.cs: class RangeT : IVariable, IEnumerable
MyTypes/MyClasses/BuiltinFunc.cs: class BuiltinFunc : IVariable, ICallable
MyTypes/MyClasses/Callable.cs: class Callable
MyTypes/MyClasses/MyClass.cs: class MyClass : ClassT
MyTypes/MyClasses/ObjectT.cs: class ObjectT : IVariable
MyTypes/MyClasses/SetT.cs: public class SetT : SortedSet<IVariable>, IVariable
MyTypes/MyClasses/ListT.cs: public class ListT : SList<ICompCloneable>, IVariable, IIndexable
MyTypes/MyClasses/PackageT.cs: public class PackageT : List<IItem>, IItem, IVariable
MyTypes/MyClasses/Method.cs: public class Method : IVariable, ICallable
MyTypes/MyClasses/BuiltinClass.cs: class BuiltinClass : ClassT
MyTypes/AccessModifier.cs: public enum AccessEnum
MyTypes/AccessModifier.cs: public class AccessModifier
MyTypes/IVariable.cs: public interface IVariable : ICompCloneable, IStringable, ITuplable
MyTypes/ISTepable.cs: interface IStepable : IVariable
MyTypes/Variable.cs: static class Variable
MyTypes/ITuplable.cs: public interface ITuplable
MyTypes/VariableFactory.cs: class VariableFactory
MyTypes/CommandNotFoundException.cs: class CommandNotFoundException : Exception
MyTypes/InfinityException.cs: class InfinityException : NumberException
MyTypes/IIndexable.cs: interface IIndexable
MyTypes/IItem.cs: public interface IItem : IVariable
MyTypes/IStringable.cs: public interface IStringable
MyTypes/ObjectContainer.cs: class ObjectContainer : List<IVariable>
MyTypes/NaNException.cs: class NaNException : NumberException
MyTypes/NotComparableException.cs: class NotComparableException : Exception
MyTypes/NumberException.cs: class NumberException : Exception
MyTypes/UndefinedException.cs: class UndefinedException : Exception
MyTypes/CircularInheritanceException.cs: class CircularInheritanceException : Exception
MyTypes/LambdaConverter.cs: static class LambdaConverter
MyTypes/ICallable.cs: public interface ICallable : IComparable
MyTypes/NoMethodException.cs: class NoMethodException : Exception
MyTypes/ModuleNotFoundException.cs: class ModuleNotFoundException : Exception
Gui/MyMenu.cs: public class MyMenu : MainMenu
Gui/FormAdapter.cs: public class FormAdapter
Gui/VisualSyntax.cs: class VisualSyntax
Gui/MainWindow.cs: class MainWindow : Form
Gui/Clickable.cs: public interface Clickable

```

```

Gui/IOPanel.cs: class IOPanel : Panel
Gui/IClickable.cs: public interface IClickable
Gui/MainPanel.cs: class MainPanel : Panel
Gui/IOBox.cs: class IOBox : RichTextBox
Gui/MyItem.cs: public class MyItem
Gui/InputGroup.cs: class InputBox : IOBox, ITexttable
Gui/OutputBox.cs: class OutputBox : IOBox, IOutputtable
MyCollections/DefaultType.cs: class DefaultType
MyCollections/WList.cs: public class WList<T> : List<T>
MyCollections/IPrintable.cs: public interface IPrintable : IEvalable, IVariable
MyCollections/EmptyArgException.cs: class EmptyArgException : Exception
MyCollections/CList.cs: public class CList<T> : WList<T> where T: IComparable
MyCollections/OStack.cs: public class OStack : WStack<object>
MyCollections/GeneralIndexer.cs: static class GeneralIndexer
MyCollections/TypeTrans.cs: public static class TypeTrans
MyCollections/SList.cs: public class SList<T> : CList<T>, ICompCloneable where T : ICompCloneable
MyCollections/ConcurrentDictionary.cs: public class ConcurrentDictionary<TKey, TValue> : Dictionary<TKey, TValue>
MyCollections/IParseable.cs: interface IParseable
MyCollections/IValuable.cs: interface IValuable
MyCollections/Pointer.cs: public class Pointer<T>
MyCollections/TokenTypes.cs: public enum TokenTypes
MyCollections/TokenTypes.cs: public static class TokenTypesExtension
MyCollections/Token.cs: public class Token
MyCollections/IEvalable.cs: public interface IEvalable
MyCollections/ICompCloneable.cs: public interface ICompCloneable : IComparable, ICloneable
MyCollections/NullArg.cs: class NullArg
MyCollections/WStack.cs: public class WStack<T> : Stack<T>, IVariable
MyCollections/General.cs: static class General
MyCollections/SortedSet.cs: public class SortedSet<T> : SortedDictionary<T,int>
MyCollections/ITexttable.cs: public interface ITexttable
Maths/NumberCalcul.cs: static class NumberCalcul
Info/Help.cs: class Help
Info/Info.cs: class Info
Controller/IRefreshable.cs: interface IRefreshable
Controller/Controller.cs: class Controller

```