THE DEVELOPER'S CONFERENCE

Trilha – Segurança

Wanderley Caloni

Sócio-Desenvolvedor da INTELITRADER

Globalcode

THE DEVELOPER'S CONFERENCE

Como Não Desenvolver Pôquer Online ou Como Explorar a Pseudo-Aleatoriedade

Globalcode

Quem sou eu? OMG!

- Segurança da Informação
  - Sistema de Controle de Usuários e Aplicações
  - Criptografia de Discos

Quem sou eu? OMG!

- Análise de Trojans
  - Engenharia Reversa
  - Crash Dump Analysis

Quem sou eu? OMG!

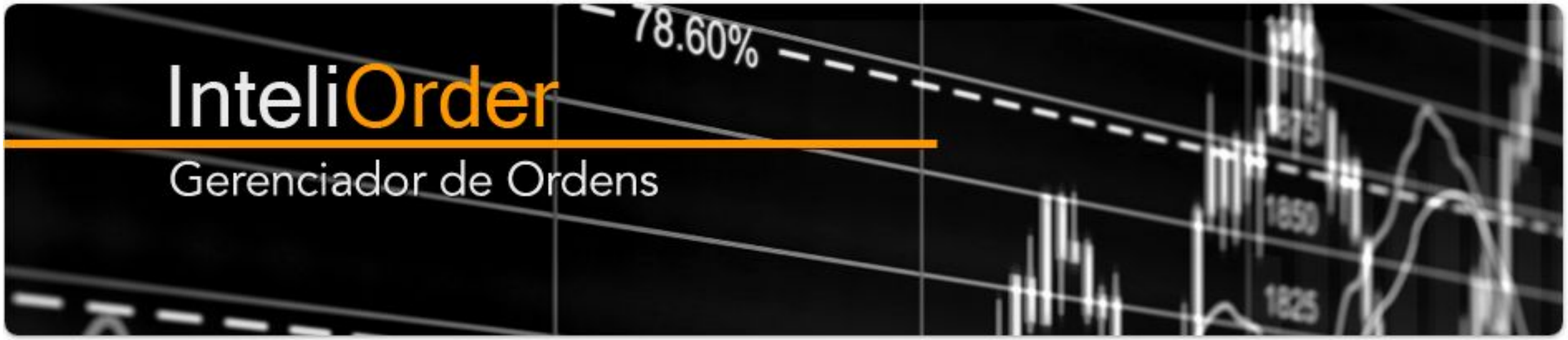
- Mercado Financeiro
  - Alto Desempenho
  - Análise de Risco

Quem sou eu? OMG!

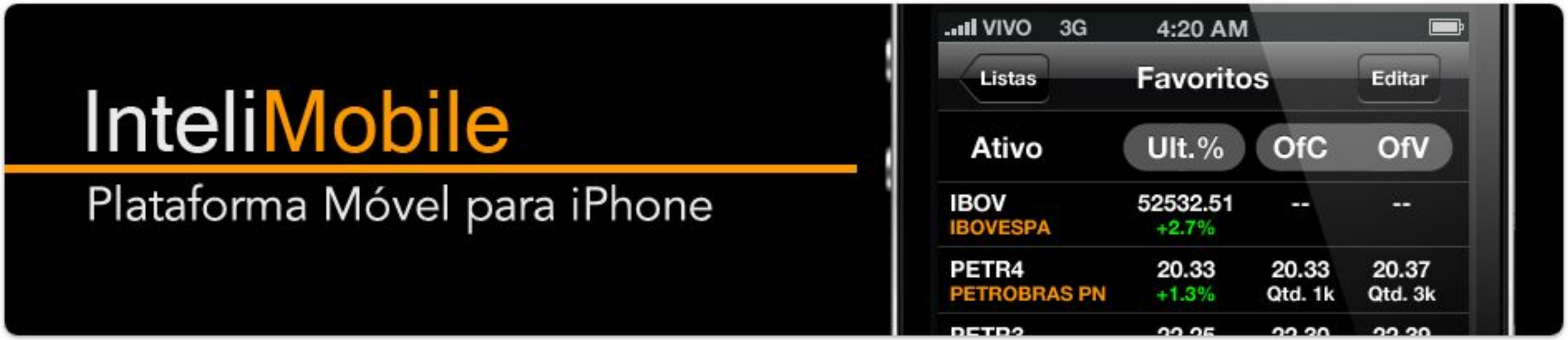
- Mercado Financeiro
  - Alto Desempenho
  - Análise de Risco
  - Algoritmos
  - Cotações
  - Mobile

INTELITRADER

Quem sou eu? OMG!



Quem sou eu? OMG!



Ativo	Ult.%	OfC	OfV
IBOV IBOVESPA	52532.51 +2.7%	--	--
PETR4 PETROBRAS PN	20.33 +1.3%	20.33 Qtd. 1k	20.37 Qtd. 3k

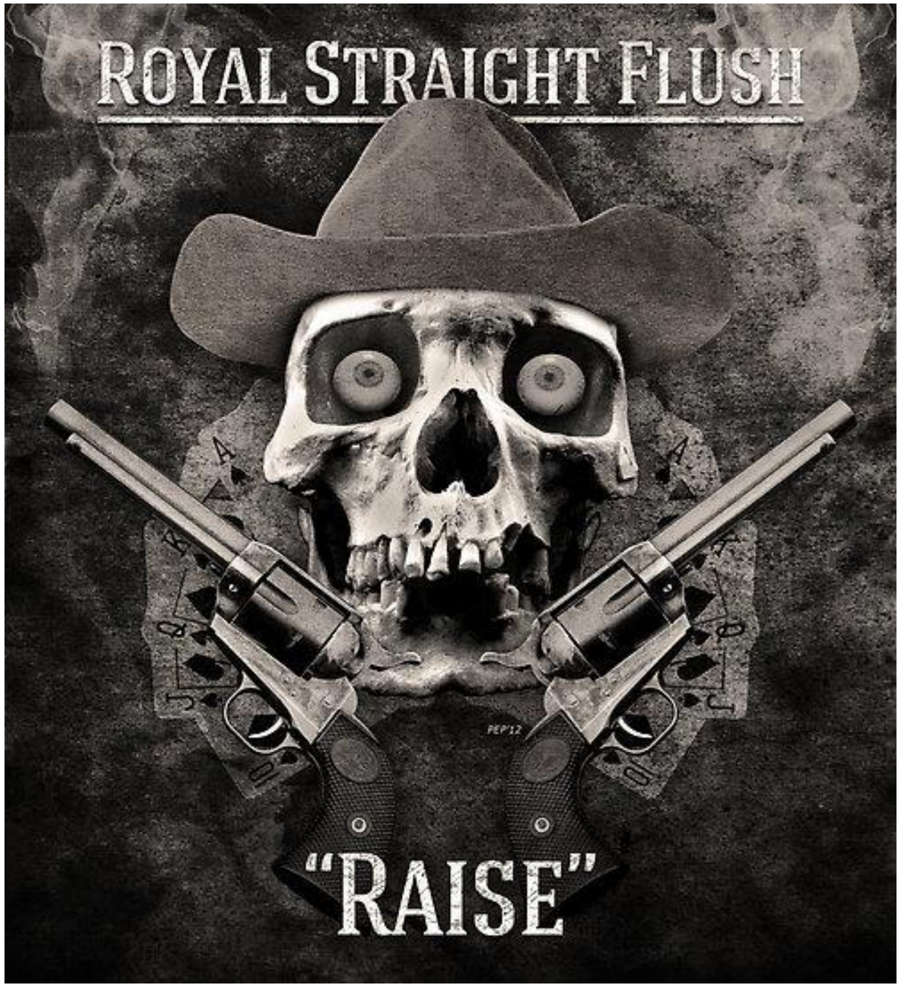
Quem sou eu? OMG!

InteliMarket

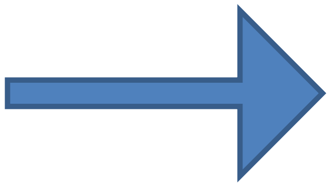
Flexibilidade em Market Data

- Balanceamento de Carga
- Certificado UMDF

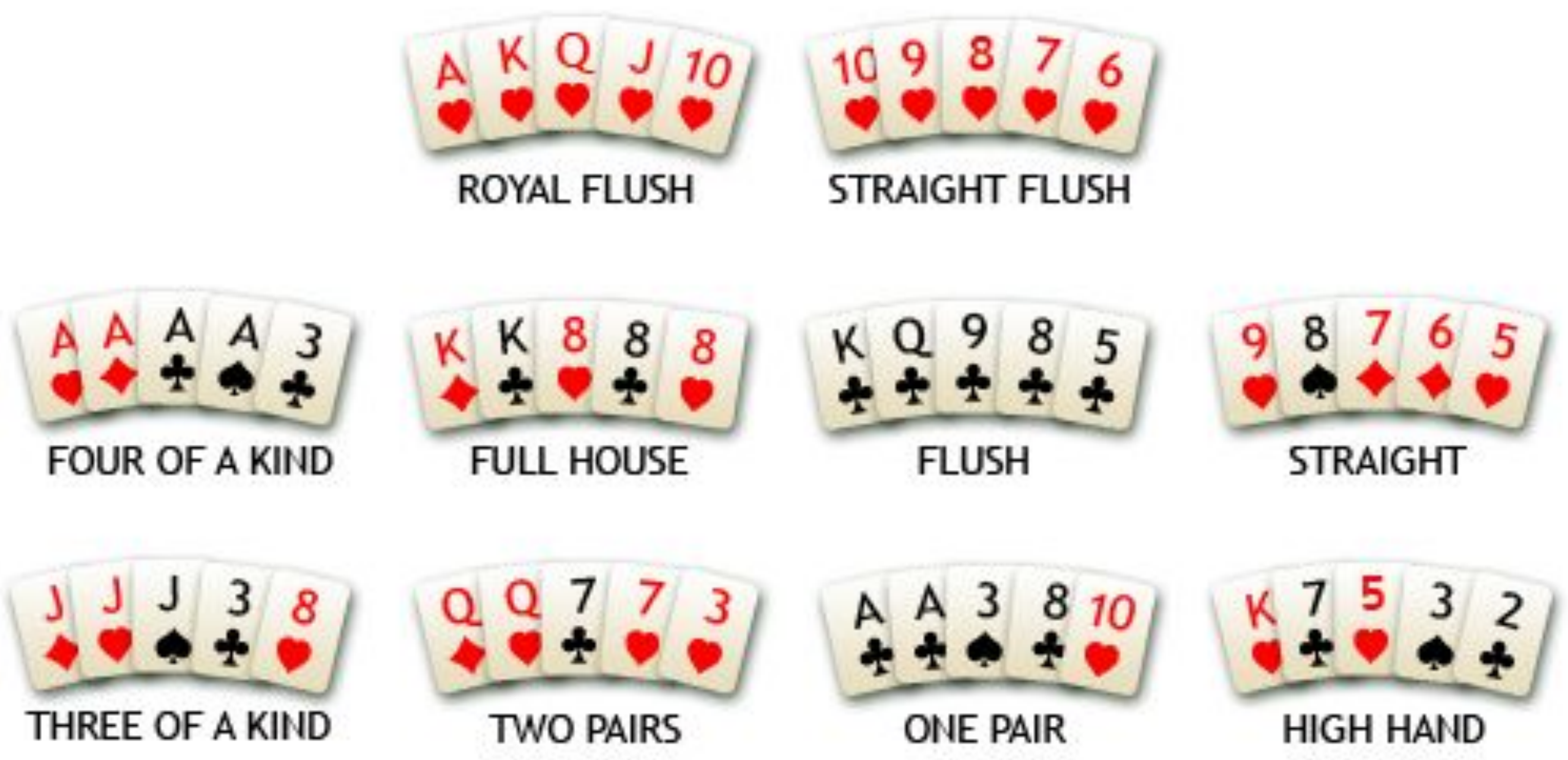
Pôquer Online



Pôquer Online



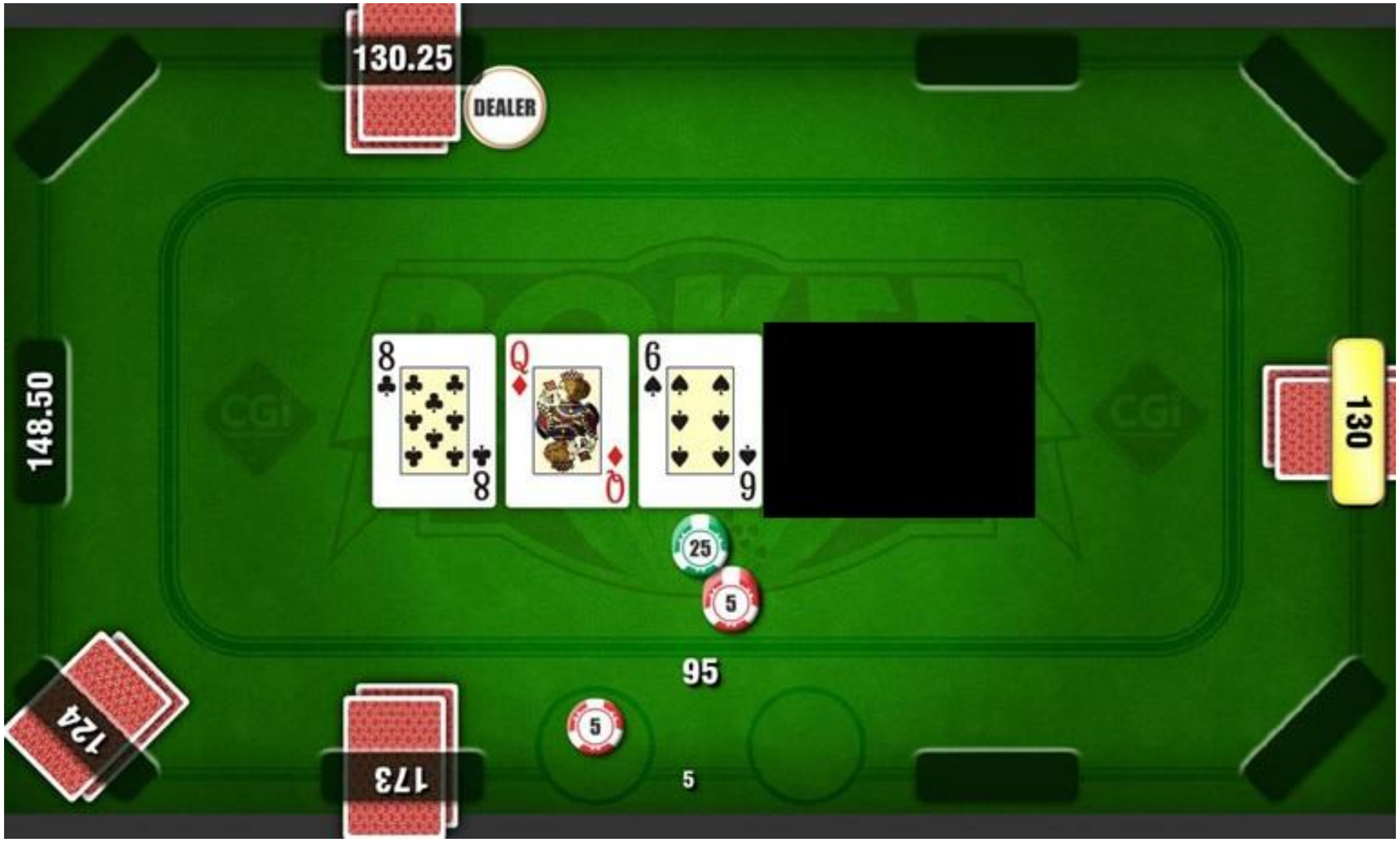
Explicar regras básicas



Explicar regras básicas

- Número de cartas: 52
  - Combinações: 52! = 8×10^{67}

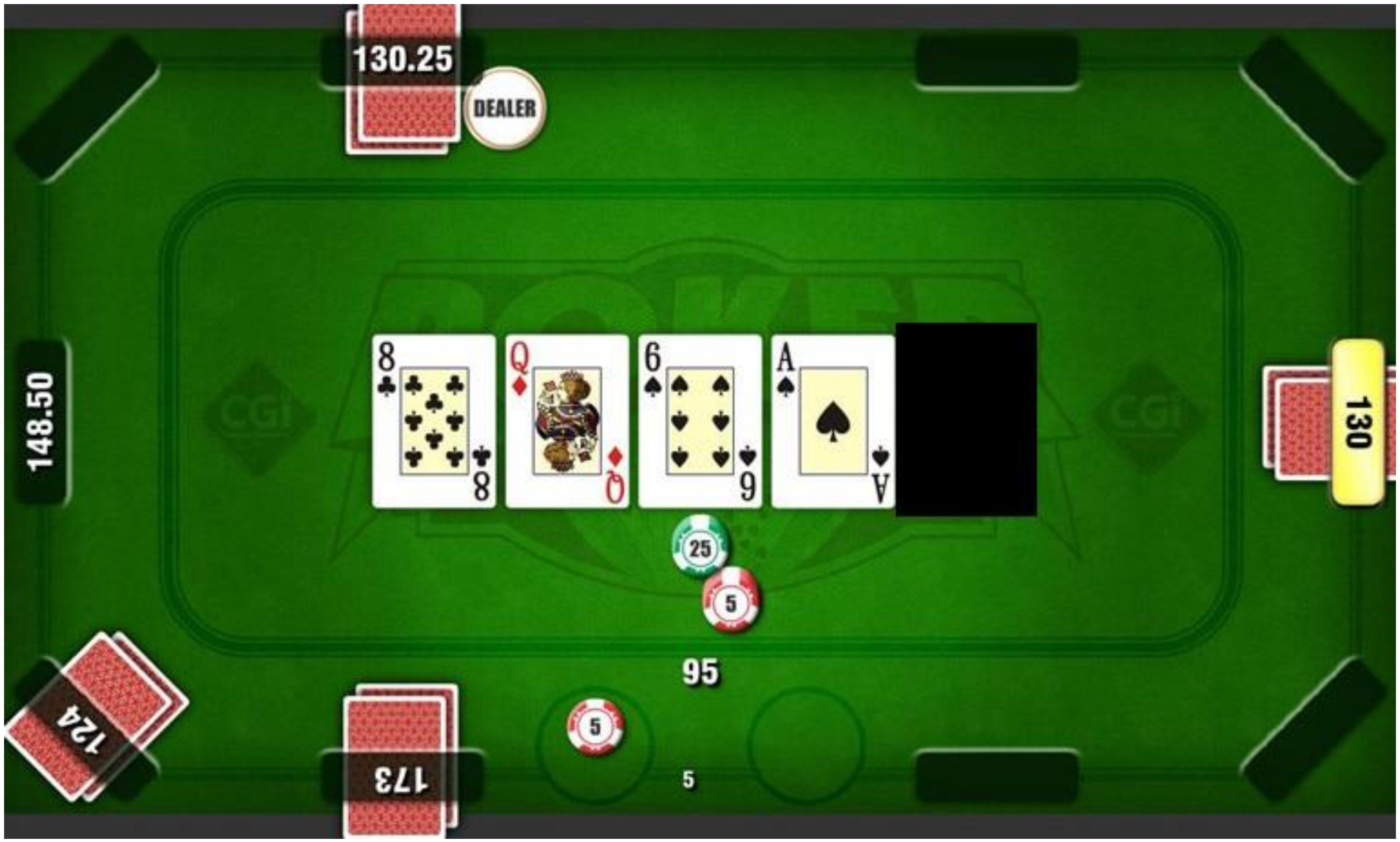
Explicar regras básicas



Explicar regras básicas

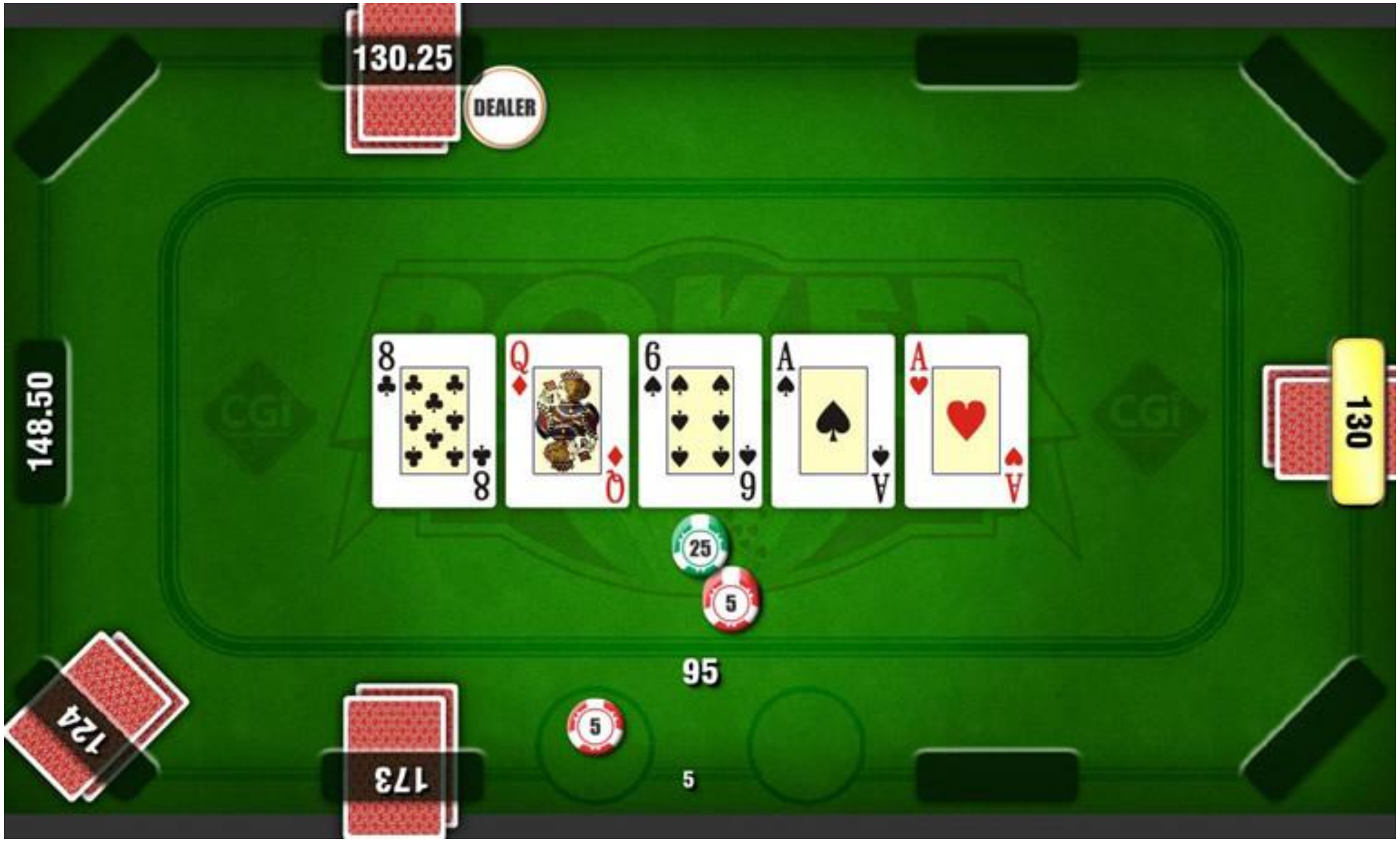
Explicar regras básicas

THE DEVELOPER'S CONFERENCE



Globalcode

Explicar regras básicas



Chupinhado Inspirado por:

THE DEVELOPER'S CONFERENCE

When Random Isn't Random Enough: Lessons from an Online Poker Exploit

February 09, 2014

Today I am going to retell a story from 1999, a story in which developers of a popular online poker platform implemented card-shuffling software with a handle of subtle but critical bugs.

Globalcode

Chupinhado Inspirado por:

LAURA HAMILTON

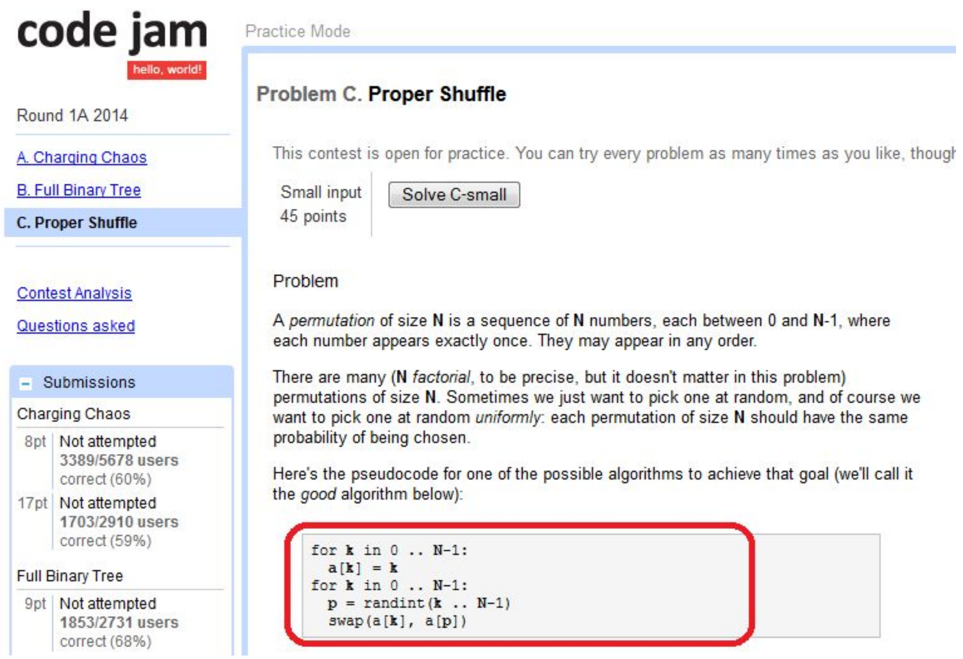
Chupinhado Inspirado por:

- http://www.lauradhamilton.com/random-lessons-online-poker-exploit

Google Code Jam 2014

THE DEVELOPER'S CONFERENCE

- https://code.google.com/codejam/contest/2984486/dashboard#s=p2
- http://www.caloni.com.br/blog/archives/poker-face



code jam
hello, world!

Practice Mode

Round 1A 2014

A. Charging Chaos
B. Full Binary Tree
C. Proper Shuffle

Contest Analysis
Questions asked

Submissions

Charging Chaos
8pt Not attempted 3389/5678 users correct (60%)
17pt Not attempted 1703/2910 users correct (59%)

Full Binary Tree
9pt Not attempted 1853/2731 users correct (68%)

Problem C. Proper Shuffle

This contest is open for practice. You can try every problem as many times as you like, thougl

Small input 45 points — Solve C-small

Problem

A *permutation* of size **N** is a sequence of **N** numbers, each between 0 and **N**-1, where each number appears exactly once. They may appear in any order.

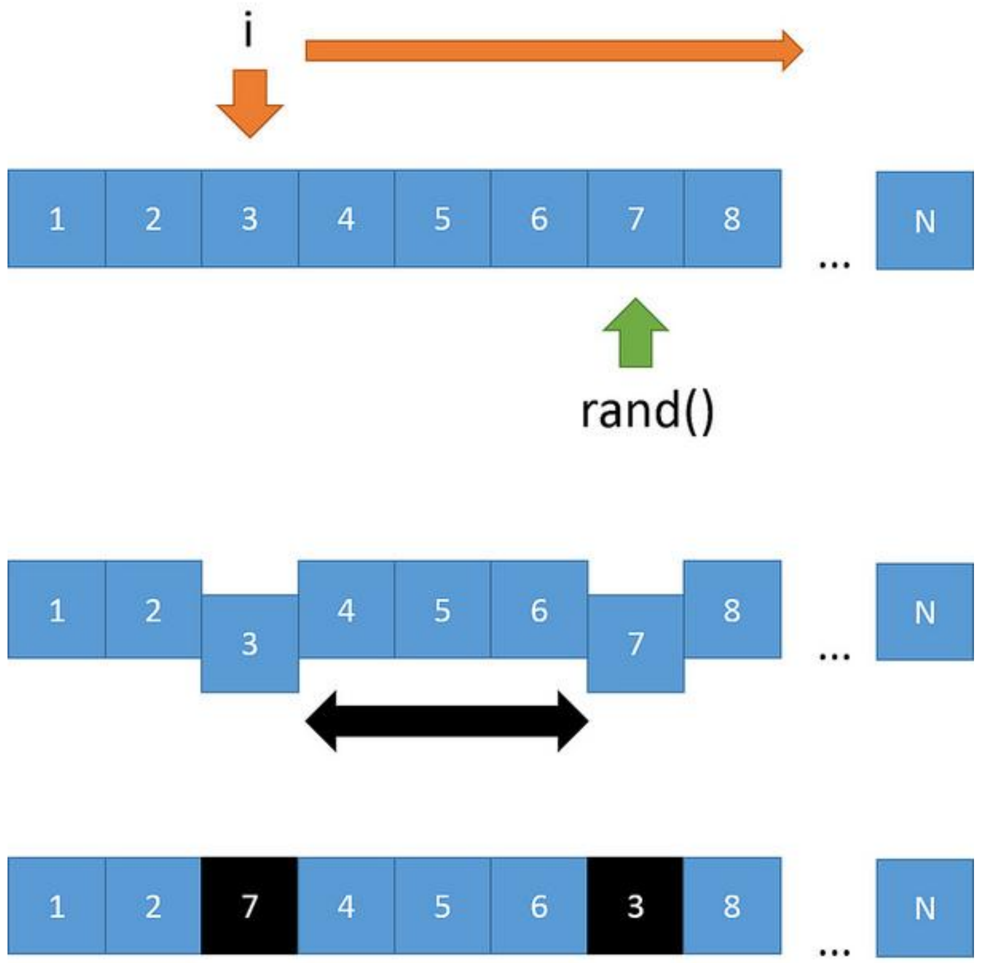
There are many (**N** *factorial*, to be precise, but it doesn't matter in this problem) permutations of size **N**. Sometimes we just want to pick one at random, and of course we want to pick one at random *uniformly*: each permutation of size **N** should have the same probability of being chosen.

Here's the pseudocode for one of the possible algorithms to achieve that goal (we'll call it the *good* algorithm below):

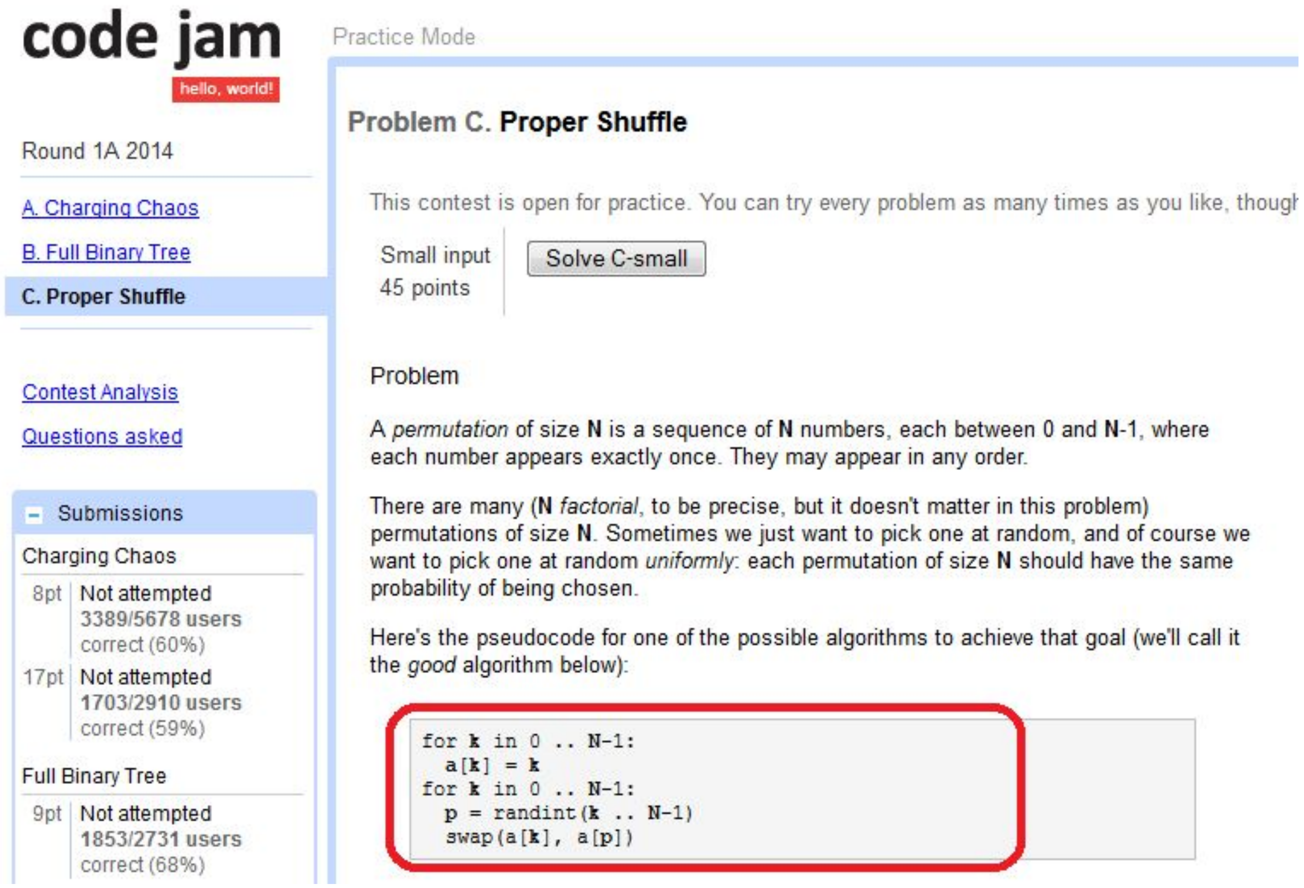
```
for k in 0 .. N-1:
  a[k] = k
for k in 0 .. N-1:
  p = randint(k .. N-1)
  swap(a[k], a[p])
```

Globalcode

Embaralhamento



Embaralhamento



code jam
hello, world!

Practice Mode

Round 1A 2014

A. Charging Chaos

B. Full Binary Tree

C. Proper Shuffle

Contest Analysis

Questions asked

Submissions

Charging Chaos

8pt	Not attempted 3389/5678 users correct (60%)
17pt	Not attempted 1703/2910 users correct (59%)

Full Binary Tree

9pt	Not attempted 1853/2731 users correct (68%)

Problem C. Proper Shuffle

This contest is open for practice. You can try every problem as many times as you like, thoug

Small input
45 points

Solve C-small

Problem

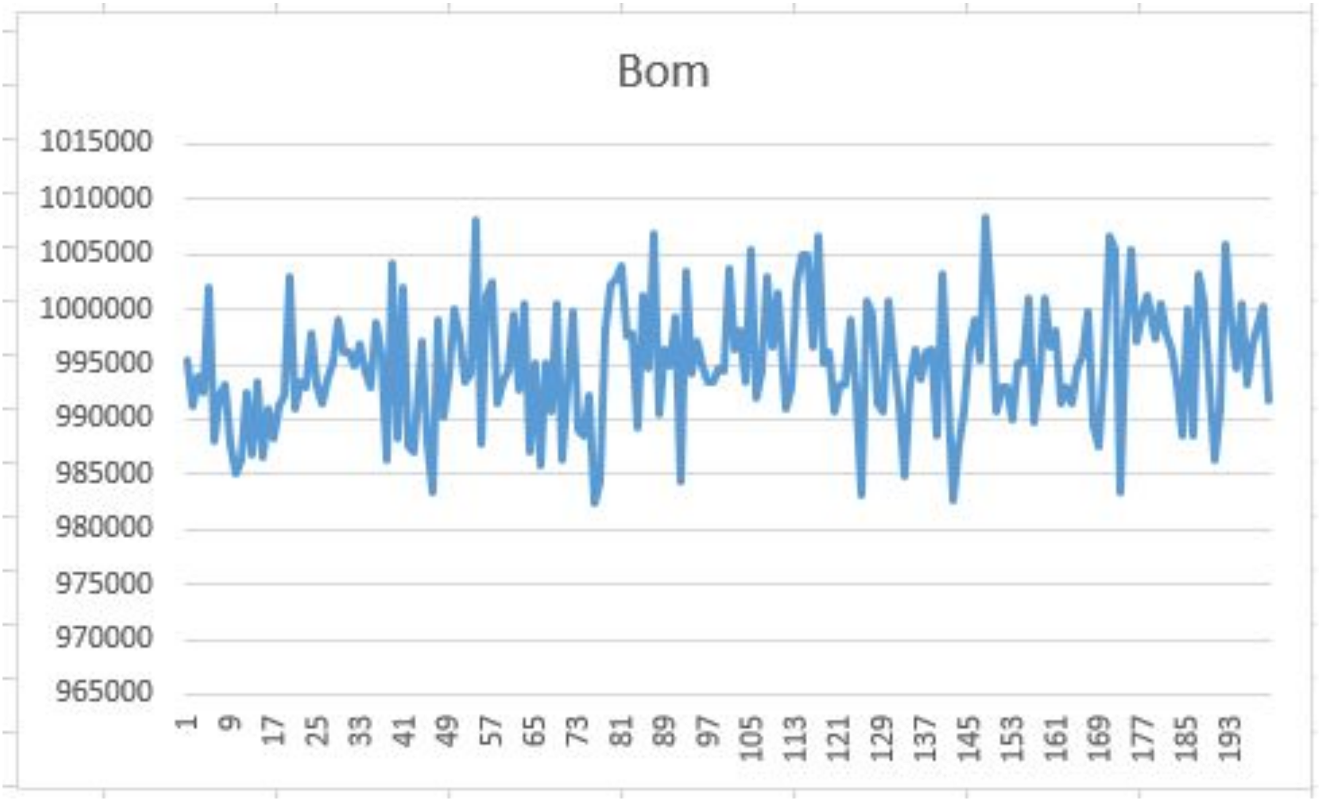
A *permutation* of size **N** is a sequence of **N** numbers, each between 0 and **N**-1, where each number appears exactly once. They may appear in any order.

There are many (**N** *factorial*, to be precise, but it doesn't matter in this problem) permutations of size **N**. Sometimes we just want to pick one at random, and of course we want to pick one at random *uniformly*: each permutation of size **N** should have the same probability of being chosen.

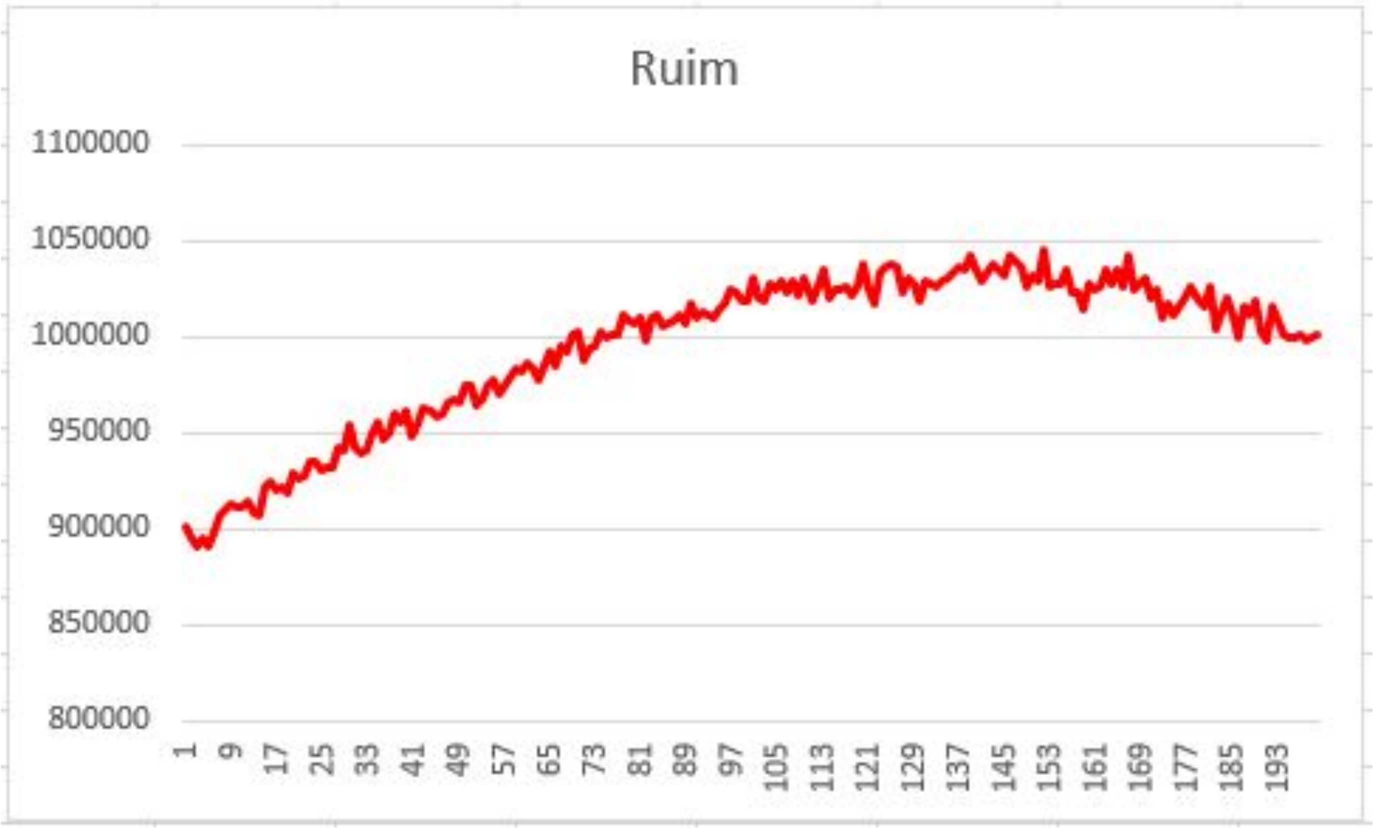
Here's the pseudocode for one of the possible algorithms to achieve that goal (we'll call it the *good* algorithm below):

```
for k in 0 .. N-1:
  a[k] = k
for k in 0 .. N-1:
  p = randint(k .. N-1)
  swap(a[k], a[p])
```

Embaralhamento



Embaralhamento



Caso Real

THE DEVELOPER'S CONFERENCE

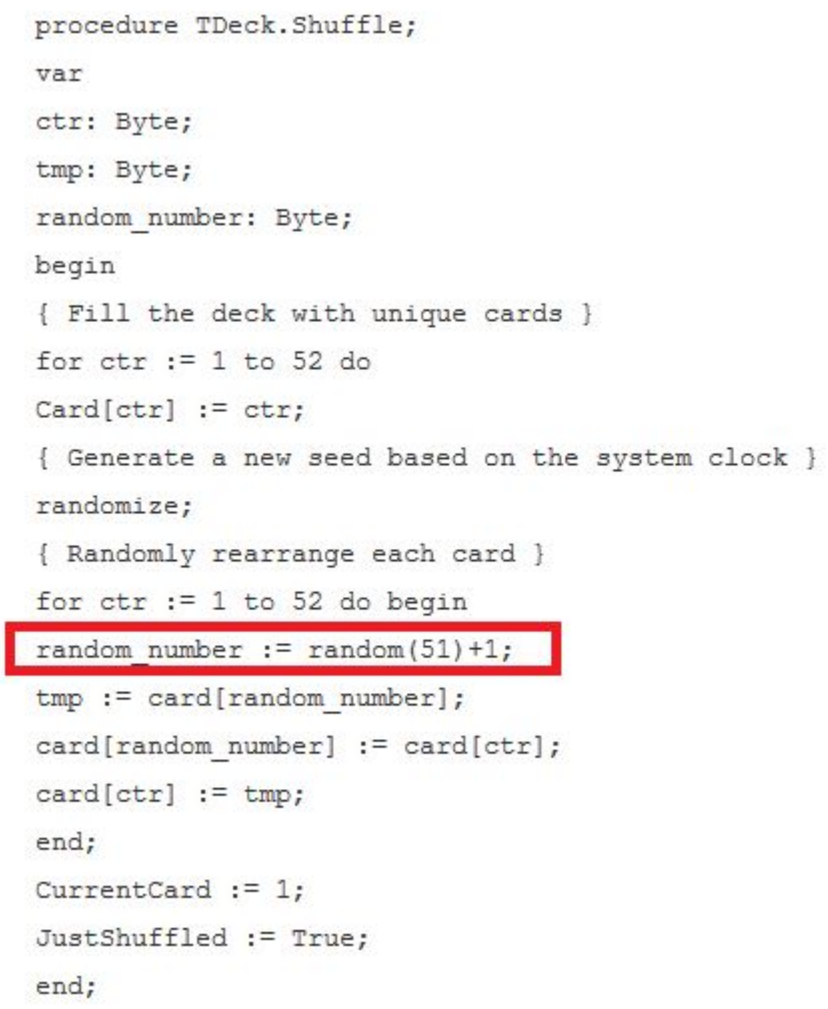
```
procedure TDeck.Shuffle;
var
ctr: Byte;
tmp: Byte;
random_number: Byte;
begin
{ Fill the deck with unique cards }
for ctr := 1 to 52 do
Card[ctr] := ctr;
{ Generate a new seed based on the system clock }
randomize;
{ Randomly rearrange each card }
for ctr := 1 to 52 do begin
random_number := random(51)+1;
tmp := card[random_number];
card[random_number] := card[ctr];
card[ctr] := tmp;
end;
CurrentCard := 1;
JustShuffled := True;
end;
```

Globalcode

Caso Real

- Falha #1: errando por um
- Falha #2: não-uniforme
- Falha #3: semente de 32 bits
- Falha #4: relógio-semente

#1: errando por um



```
procedure TDeck.Shuffle;
var
ctr: Byte;
tmp: Byte;
random_number: Byte;
begin
{ Fill the deck with unique cards }
for ctr := 1 to 52 do
Card[ctr] := ctr;
{ Generate a new seed based on the system clock }
randomize;
{ Randomly rearrange each card }
for ctr := 1 to 52 do begin
random_number := random(51)+1;
tmp := card[random_number];
card[random_number] := card[ctr];
card[ctr] := tmp;
end;
CurrentCard := 1;
JustShuffled := True;
end;
```

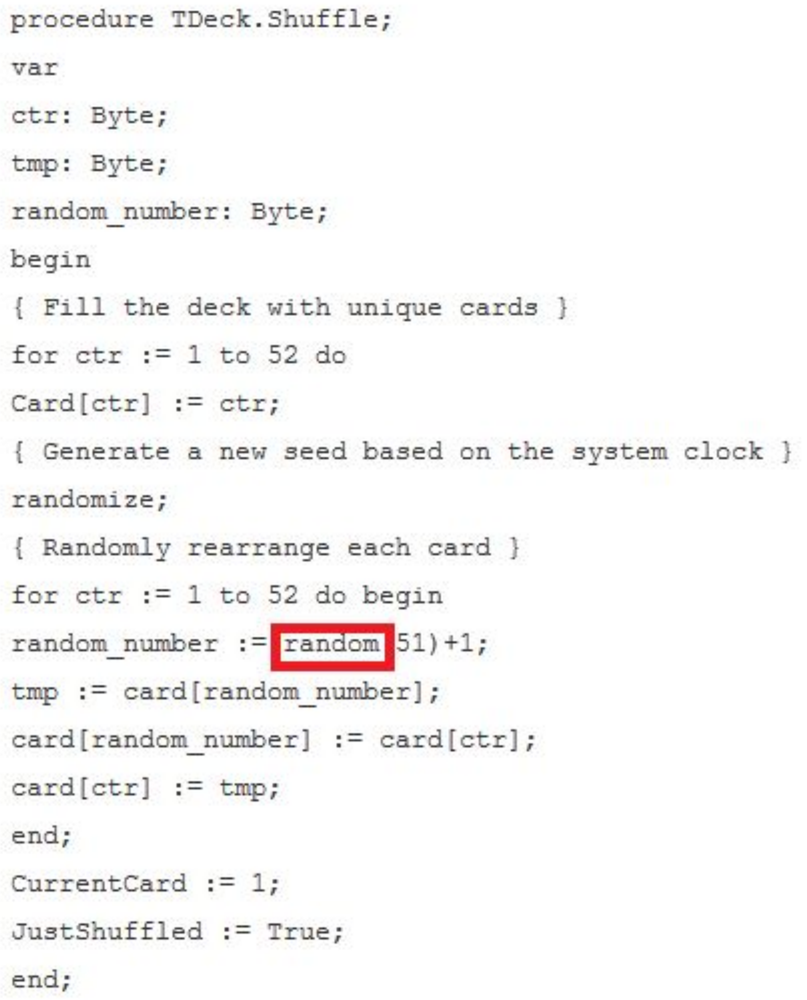
#2: não-uniforme

```
procedure TDeck.Shuffle;
var
ctr: Byte;
tmp: Byte;
random_number: Byte;
begin
{ Fill the deck with unique cards }
for ctr := 1 to 52 do
Card[ctr] := ctr;
{ Generate a new seed based on the system clock }
randomize;
{ Randomly rearrange each card }
for ctr := 1 to 52 do begin
random_number := random(51)+1;
tmp := card[random_number];
card[random_number] := card[ctr];
card[ctr] := tmp;
end;
CurrentCard := 1;
JustShuffled := True;
end;
```

```
for k in 0 .. N-1:
  a[k] = k
for k in 0 .. N-1:
  p = randint(k .. N-1)
  swap(a[k], a[p])
```

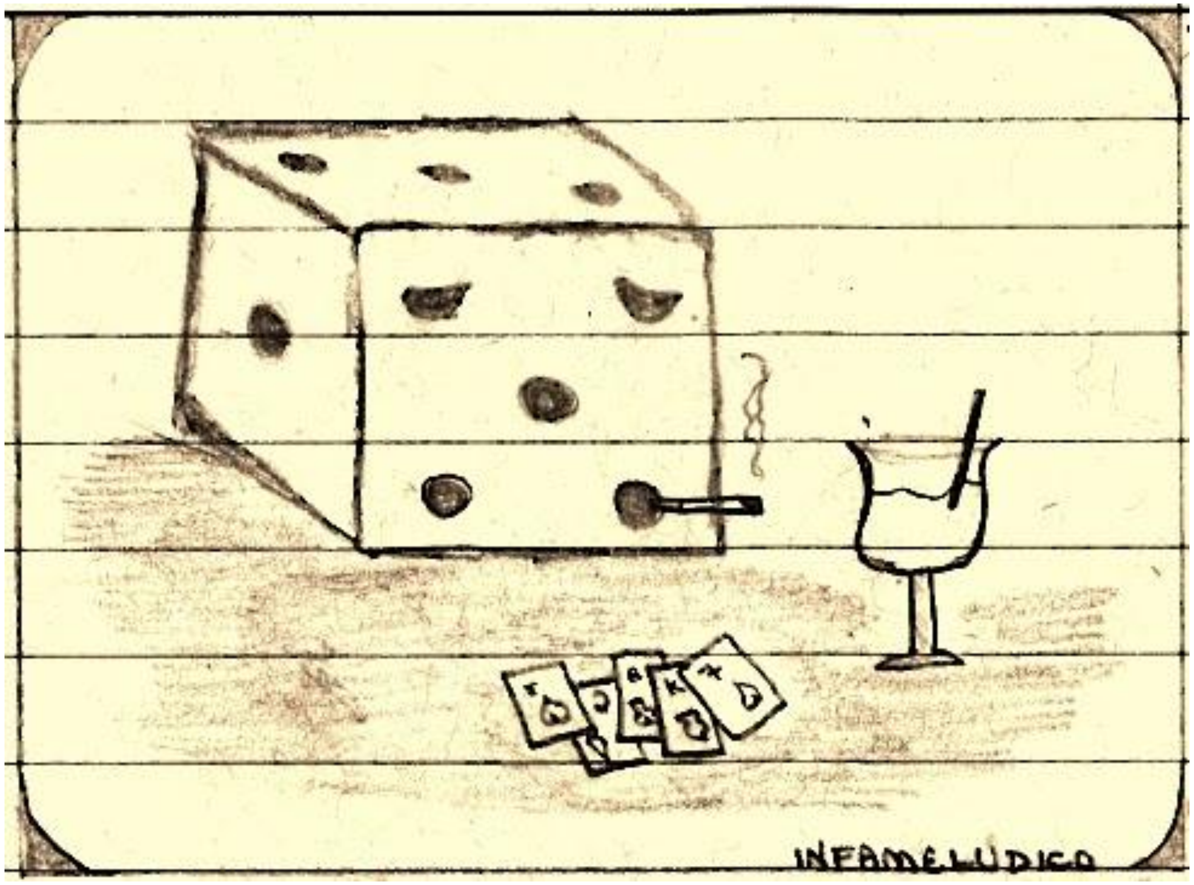
```
for k in 0 .. N-1:
  a[k] = k
for k in 0 .. N-1:
  p = randint(0 .. N-1)
  swap(a[k], a[p])
```

#3: semente de 32 bits



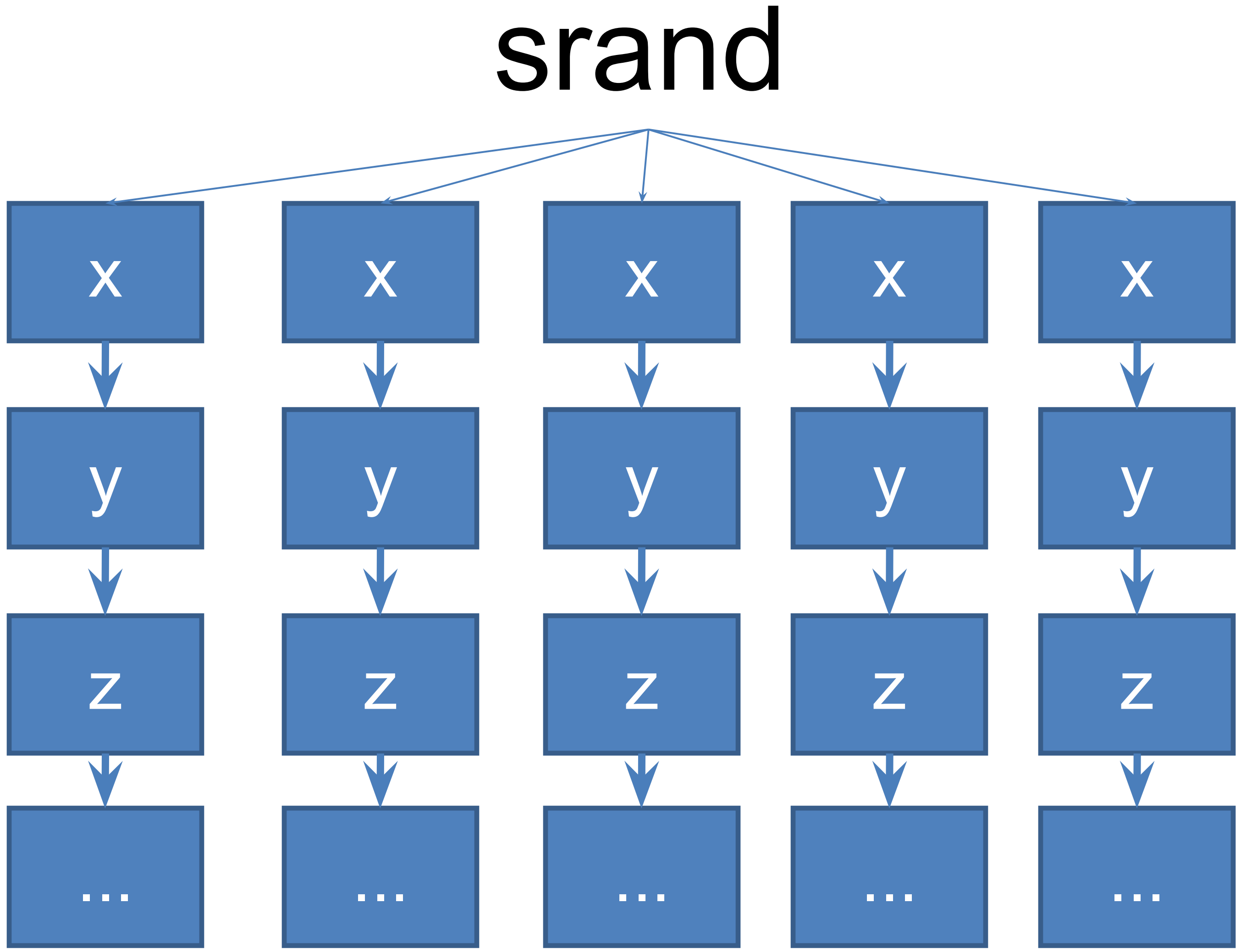
```
procedure TDeck.Shuffle;
var
ctr: Byte;
tmp: Byte;
random_number: Byte;
begin
{ Fill the deck with unique cards }
for ctr := 1 to 52 do
Card[ctr] := ctr;
{ Generate a new seed based on the system clock }
randomize;
{ Randomly rearrange each card }
for ctr := 1 to 52 do begin
random_number := random(51)+1;
tmp := card[random_number];
card[random_number] := card[ctr];
card[ctr] := tmp;
end;
CurrentCard := 1;
JustShuffled := True;
end;
```

#3: semente de 32 bits



rand

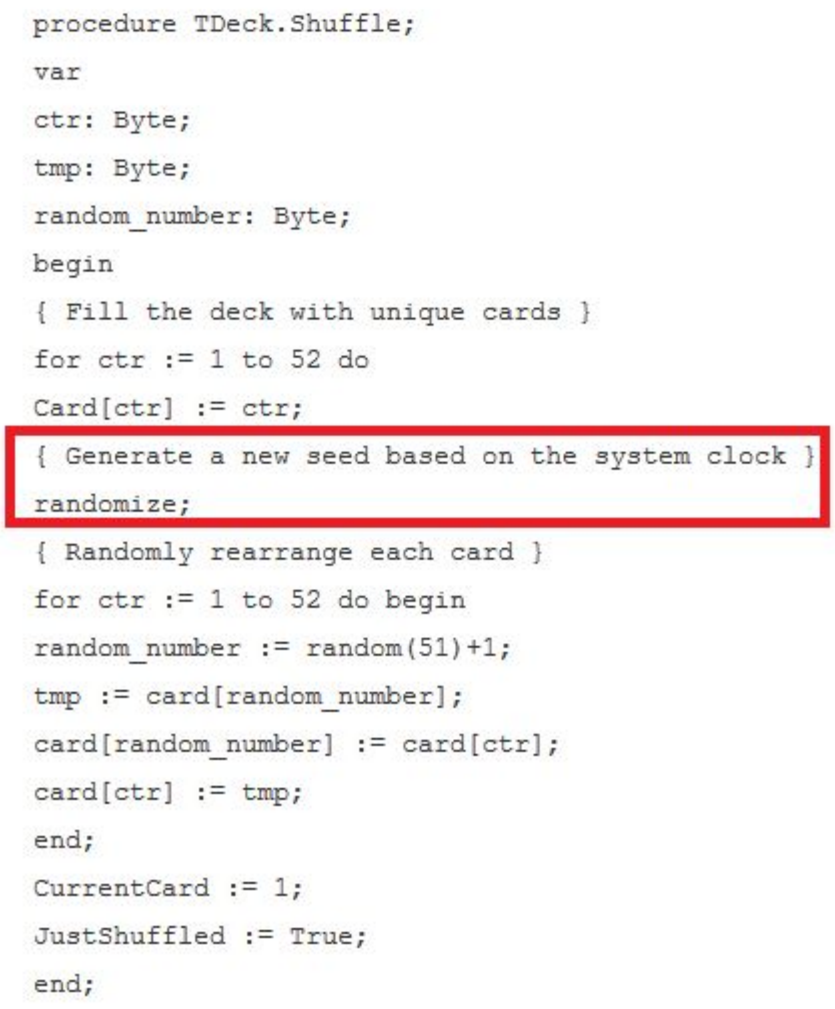
#3: semente de 32 bits



#4: relógio-semente

#4: relógio-semente

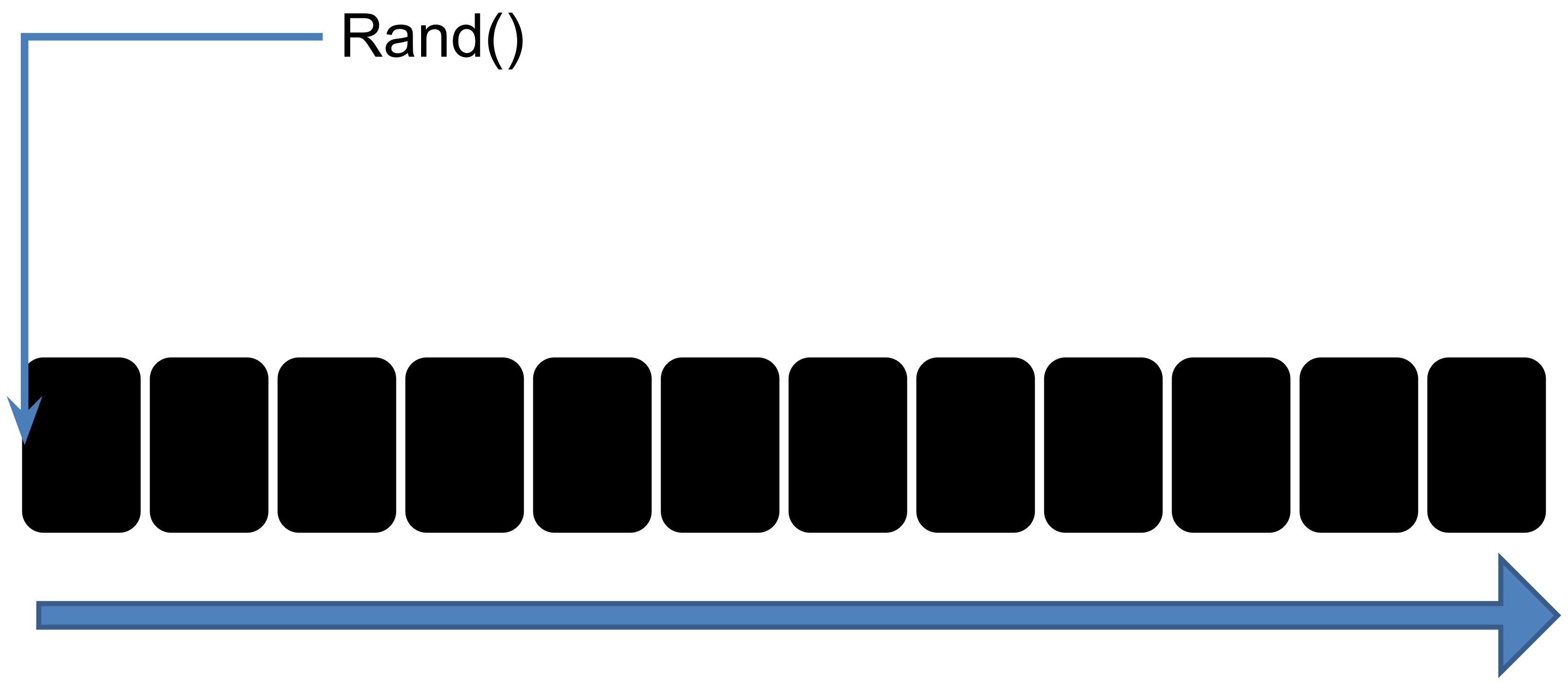
THE DEVELOPER'S CONFERENCE



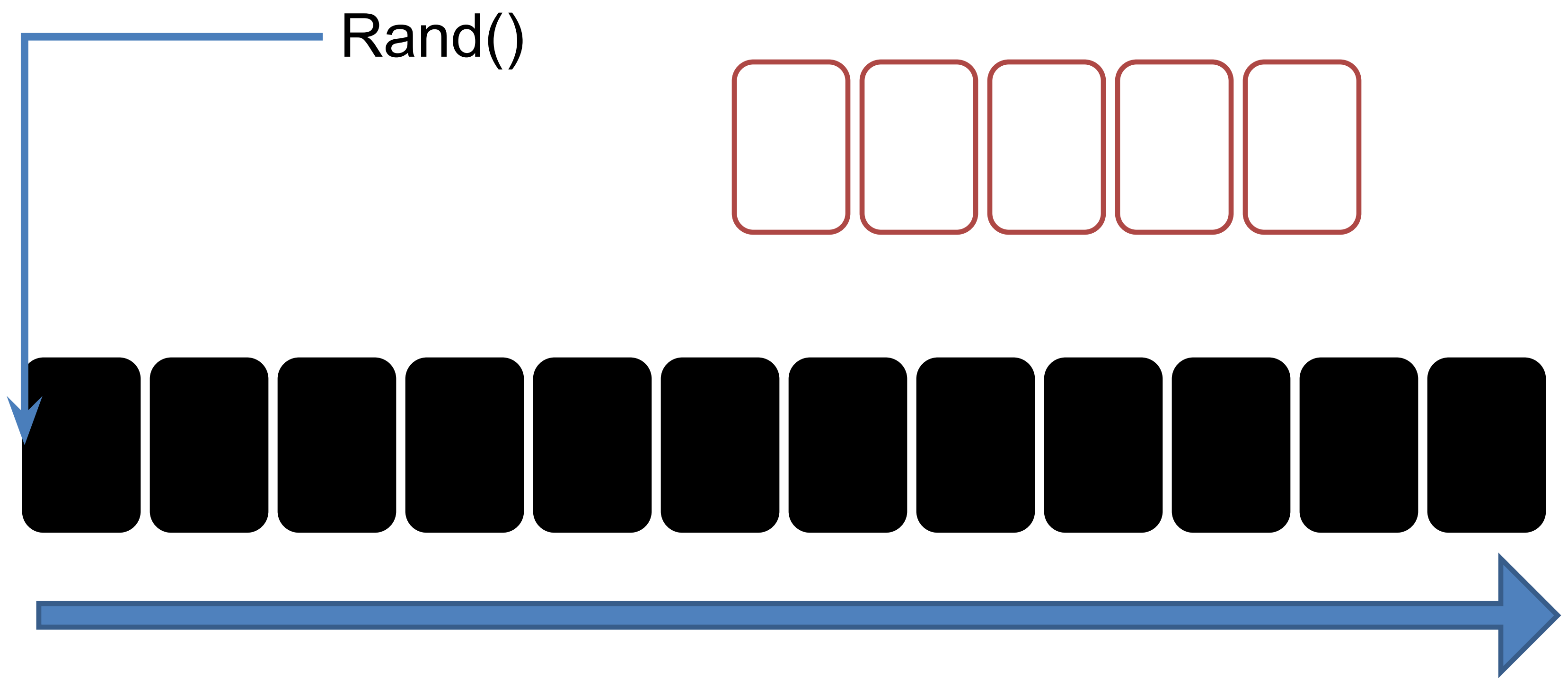
```
procedure TDeck.Shuffle;
var
ctr: Byte;
tmp: Byte;
random_number: Byte;
begin
{ Fill the deck with unique cards }
for ctr := 1 to 52 do
Card[ctr] := ctr;
{ Generate a new seed based on the system clock }
randomize;
{ Randomly rearrange each card }
for ctr := 1 to 52 do begin
random_number := random(51)+1;
tmp := card[random_number];
card[random_number] := card[ctr];
card[ctr] := tmp;
end;
CurrentCard := 1;
JustShuffled := True;
end;
```

Globalcode

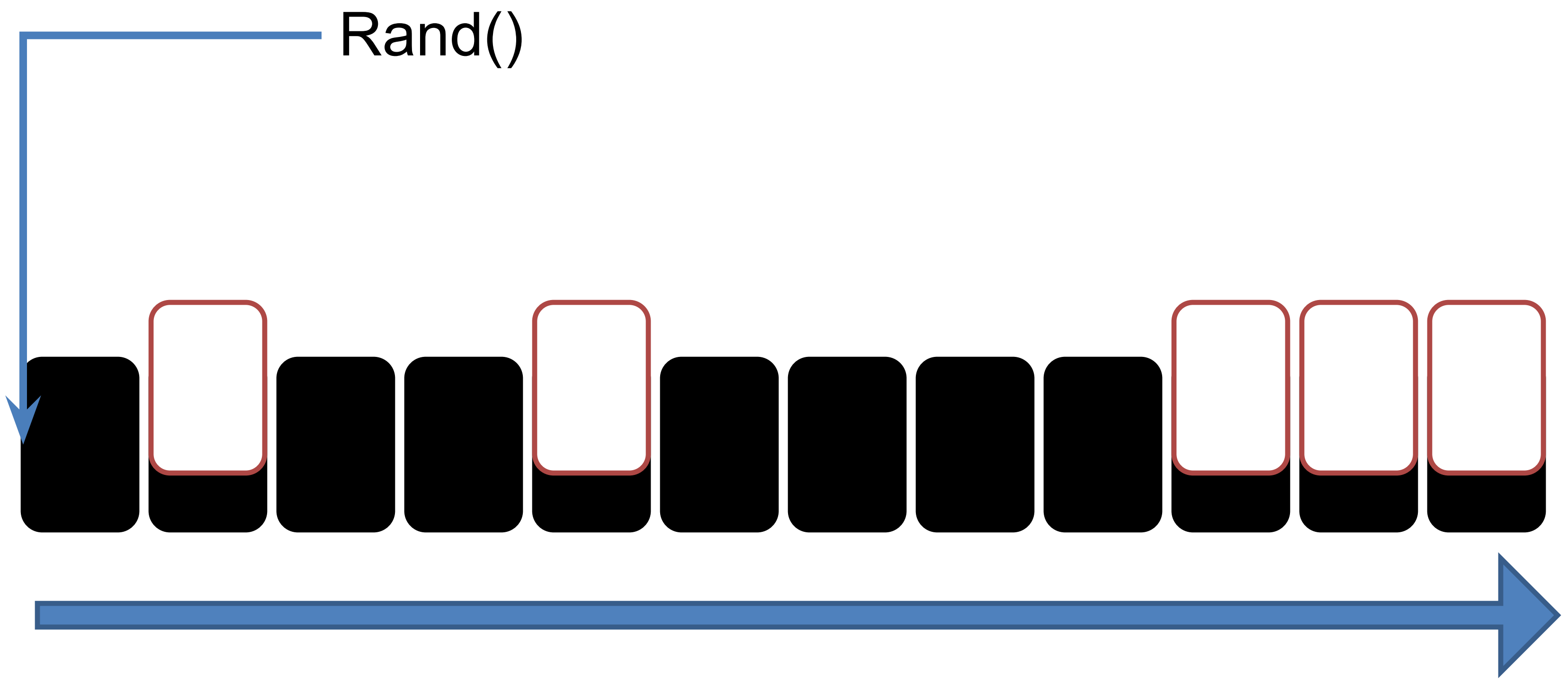
Exploit



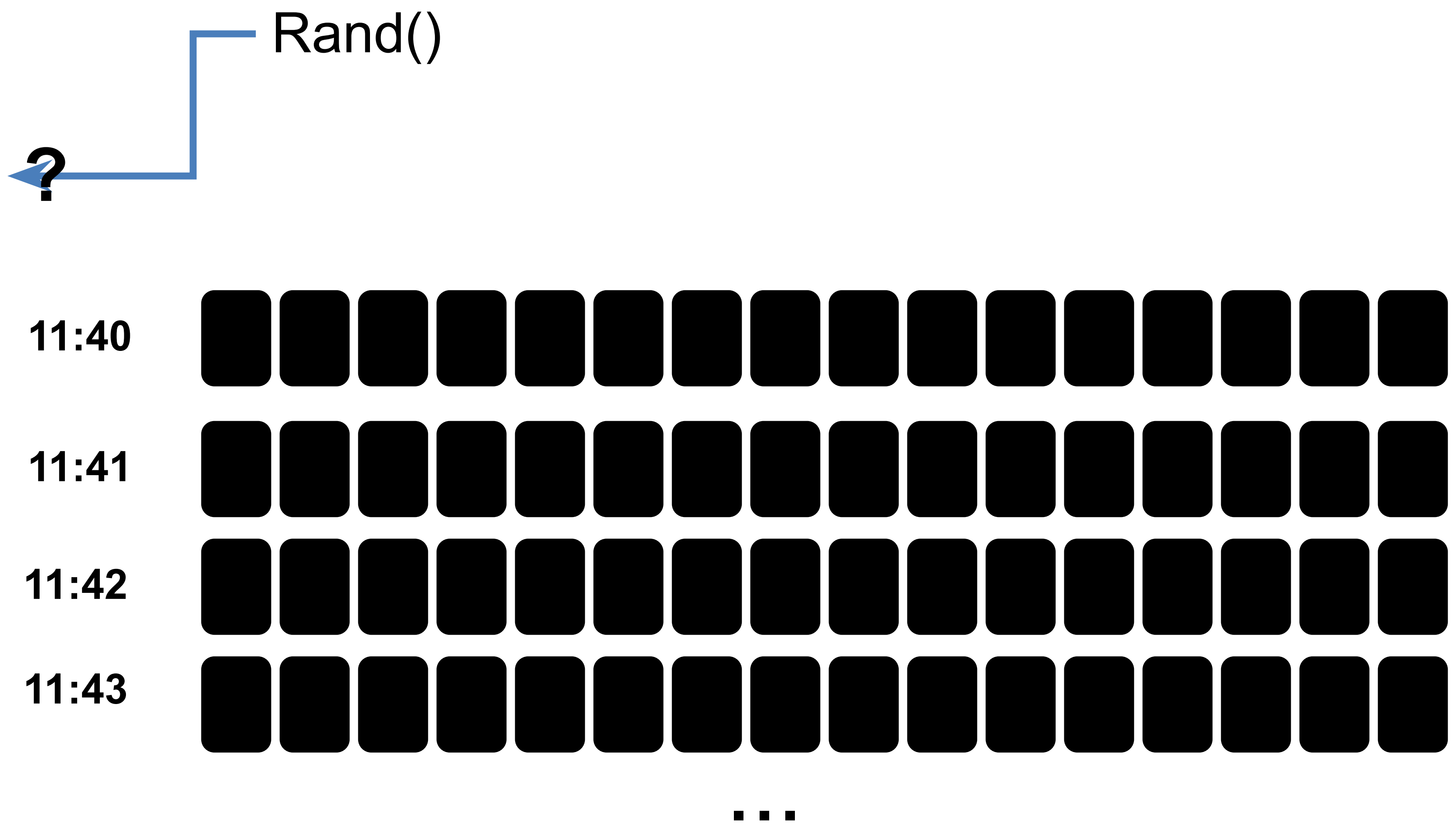
Exploit



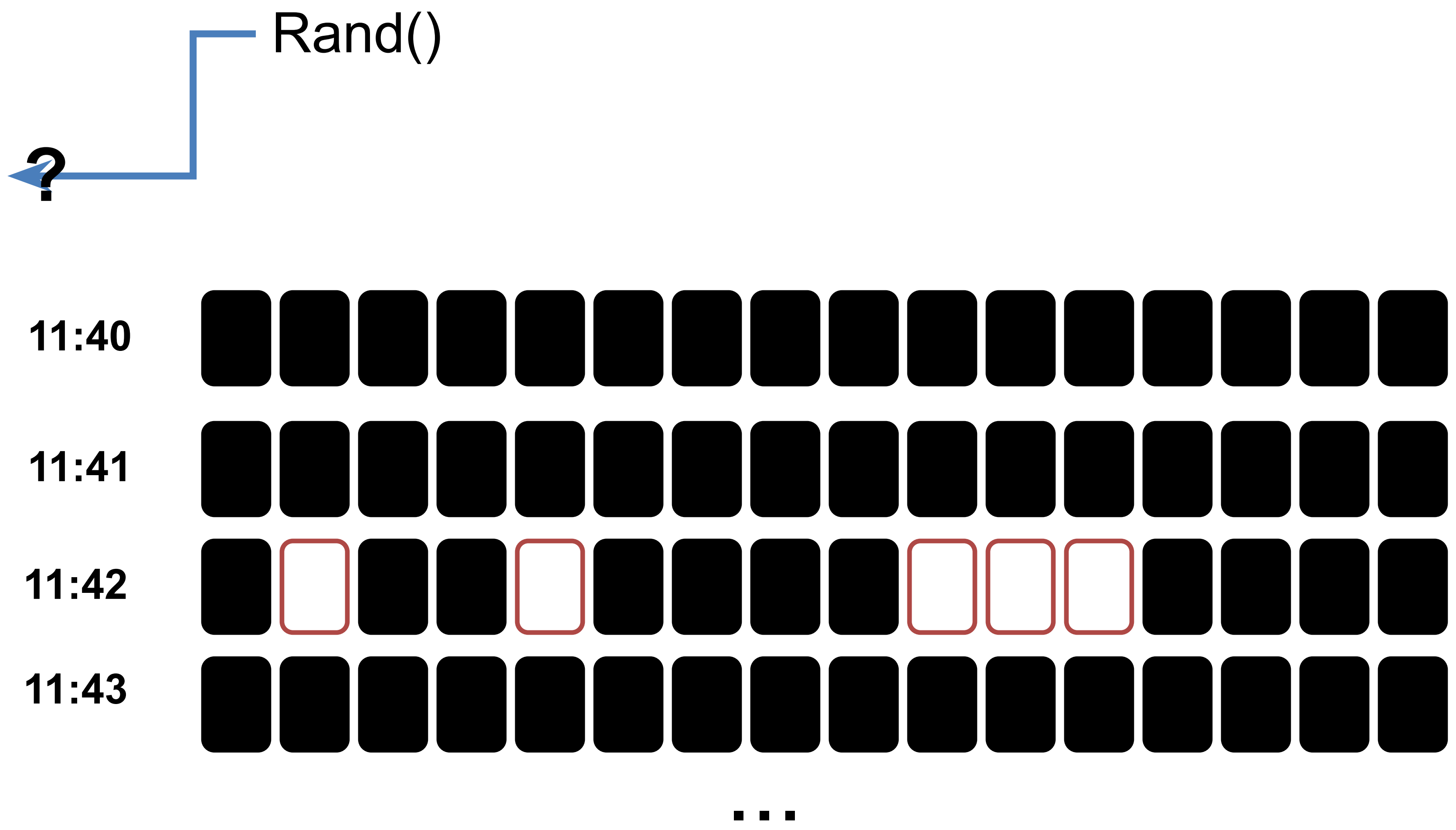
Exploit



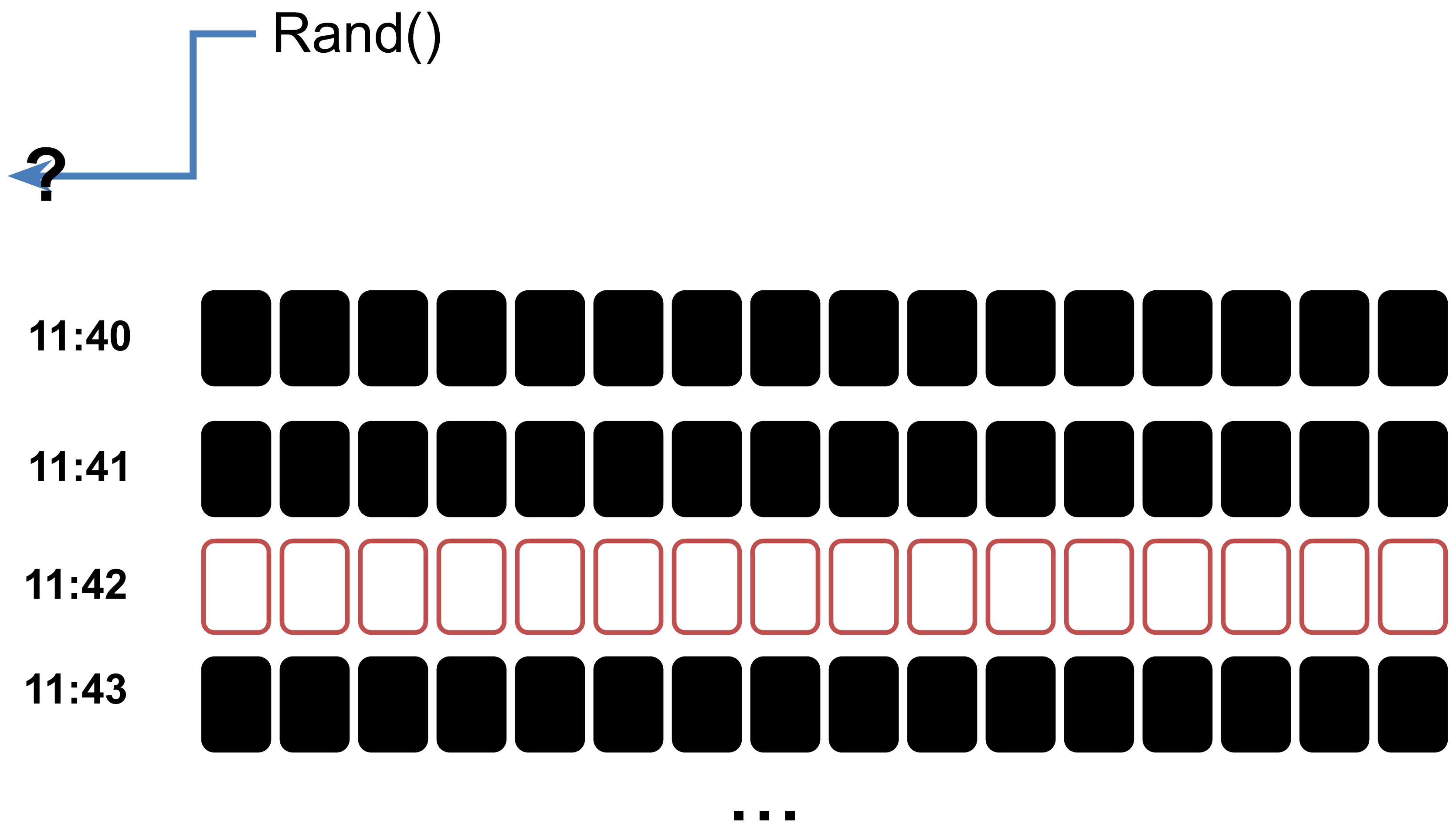
Exploit



Exploit



Exploit



And in the end....

THE DEVELOPER'S CONFERENCE

Perguntas? Eu tenho várias.

e-mail

$$\overbrace{\textbf{wanderley@caloni.com.br}}$$

- **wanderley@caloni.com.br**

twitter

site

Globalcode

And in the end....

INTELITRADER .com.br