EZFIN-DSL

Lógica da Computação 2023.1

Paulo Souza Chade

Insper

Motivação

Inspiracões

- C, Solidity

Resumo

1. Linguagem para tratar de transferências financeiras
2. Facilitar o uso para programas/pessoas no contexto bancário
3. Oferece uma programação Turing Completa - possibilita a adição de códigos complexos sobre a infraestrutura já construída

Características

Resumo

1. Linguagem tipada
2. Precisão: não admite uso de pontos flutuantes (números não inteiros). Isso pois é feita para tratar transações
3. Possibilita uso de condicionais (if), loops (loop) e declaração de funções (function)
4. As funções específicas são create, deposit, withdraw e transfer
5. Implementada utilizando **llvm**

Limitacões

- Funções: Não admite recursão
- Funções: Apenas podem ter um statement de retorno. E o mesmo deve estar no escopo principal da função (fora de if/loop)
- Condicional: Apenas implementa IF, não implementa ELSE nem ELIF
- Tipagem: nesta primeira versão apenas trata o tipo INT
- Não implementa comparações como OR e AND

obs: Após rodar o compilador, ele pode dar segmentation-fault. Contudo, todo o código é rodado, e os outputs são visíveis. Portanto não influencia no compilador

Exemplo: Turing

Exemplo de funcionalidades que oferecem uma língua Turing completa

Função externa para poder invocar "printf" do c (questões de debug e visualizar output)

Exemplo de declaração de função: "function" + tipo retorno + nome + argumentos + bloco

Exemplo de loop condicional: "loop" + condicão + bloco

Exemplo de condicional if: "If" + condicão + bloco

Exemplo retorno (está no escopo principal da função): "return" + expressão

Exemplo de declaração de variável e atribuicão de valor: tipo + nome + "=" + expressão

Exemplo de chamada de função: nome + argumentos

```
extern void printi(int val)


function int test(int c, int h) {
    if(c > h) {

        loop(c > h + 10) {
            h = h + 1
        }
    }
    if(c == h) {
        h = 1000/2
        int j = c/2
        if(j == 0) {
            h = 325
        }
    }
    return h
}

int a = test(10000, 1)
printi(a)
int b = test(1, 1)
printi(b)
```

Output

```
Running code...
9990
325
```

Exemplo: Geração de código intermediário LLVM

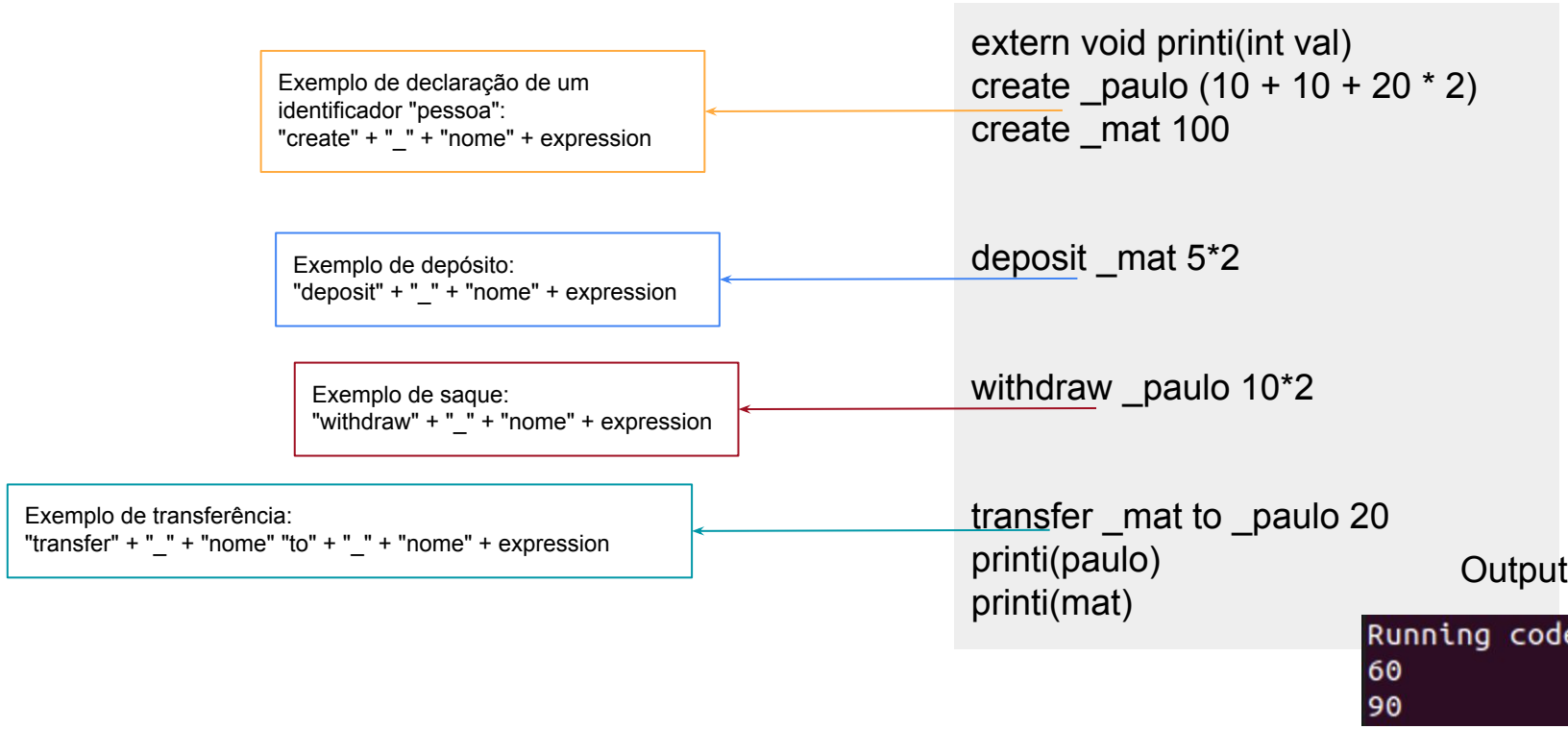
```
int a = 2

if(10 > 9) {
    a = 20
    if(10 > 9) {
        a = 100
    }
}
printi(a)
```



```
define internal i64 @main() {
entry:
  %a = alloca i64, addrspace(4)
  store i64 2, i64 addrspace(4)* %a
  %0 = icmp sgt i64 10, 9
  %1 = icmp ne i1 %0, false
  br i1 %1, label %then, label %ifmerge2

then:                                             ; preds = %entry
  store i64 20, i64 addrspace(4)* %a
  %2 = icmp sgt i64 10, 9
  %3 = icmp ne i1 %2, false
  br i1 %3, label %then1, label %ifmerge

then1:                                            ; preds = %then
  store i64 100, i64 addrspace(4)* %a
  br label %ifmerge

ifmerge:                                          ; preds = %then1, %then
  br label %ifmerge2

ifmerge2:                                         ; preds = %ifmerge, %entry
  %a3 = load i64 addrspace(4)*, i64 addrspace(4)* %a
  call void @printi(i64 addrspace(4)* %a3)
}
```

Exemplo: DSL

Exemplo de funcionalidades especiais da língua



Exemplo: Geral

```
function int has_balance(int ident, int bal) {
    int ret = 0
    if(ident >= bal) {
        ret = 1
    }
    return ret
}

int a = 7
int b = a * 10

create _paulo (b * 2)
create _mat (a)

int bal_paulo = has_balance(paulo, b)
if(bal_paulo == 1) {
    transfer _paulo to _mat (b/2)
}
```


Verifica se pessoa tem balança antes de transferir