APS-Linguagem de Caixa-Eletrônica Automática

Daniel Juho Kim

Objetivo da linguagem

Essa é uma linguagem de programação que pode ser usada para fazer simulações de um caixa eletrônico automático de um mercadinho de um prédio, onde a partir dele, podemos simular como seria o processo de um residente fazer compras no mercadinho do prédio e como seria feito o controle dos produtos no depósito/armazém do prédio

Motivo para a escolha desse tema

No meu condomínio, tem um mercadinho que fica no térreo, usado frequentemente pelos residentes, pela curiosidade de saber como isso funciona e pela busca de um tema para esse trabalho, conversei com alguns funcionários responsáveis pelo gerenciamento do mercadinho, assim como busquei ver como funciona uma caixa automática e processos de compra com maquininha

Tendo conversado com os funcionários, vi que para o gerenciamento do mercadinho, eles procuram determinados produtos comprados pelos moradores e os mantém guardados no armazém/depósito, e com a caixa automática, ela interage com o banco de dados que tem informações sobre os produtos no armazém e calcula a quantidade de unidades de um produto comprado pelo morador com a quantidade desse produto no armazém

Além disso, verifiquei como é o fluxo do caixa, que ao receber um QR-Code, busca no banco de dados o produto correspondente ao produto e mantém esse produto guardado em uma memória temporária que ao prosseguir para o processo de compra, caso o pagamento tenha sido aprovado, percorre e apresenta todos os produtos dessa memória temporária, mostrando a quantidade de unidades compradas e o valor total da compra

Fluxo de Compilação/Interpretação da VM

parser.y + lexer.l → compilador, é gerado um arquivo executável que faz a análise sintática e léxica

compilado → Recebe um arquivo.jota que vai ter a linguagem de programação estruturada e vai passar pelo processo de compilação, gerando um arquivo.asm

VM → Recebe um arquivo.asm e faz a análise semântica da linguagem, além de também processar as instruções geradas no arquivo durante a compilação

Instruções e formatos sintáticos da linguagem

1. INICIAR

Para iniciar o processo de execução da linguagem, precisamos usar a instrução INICIAR, que vai sinalizar o início do programa

2. FINALIZAR

Para o fim do programa, é necessário ter a instrução FINALIZAR que vai sinalizar o fim do programa, caso seja usado antes da linha de PAGAMENTO, vai CANCELAR a compra, caso contrário, vai perguntar se vai querer imprimir a nota fiscal

3. PRODUTO

Registrador que é usado para guardar informações sobre um determinado produto com um QR-Code de preço X

Pode ter seu preço consultado ao usar: CONSULTAR qr-code-produto

4. DEPOSITO

Memória usada para guardar X unidades de Y produtos, onde primeiro precisa ser criado:

CRIAR_DEPOSITO depositoCondominio7 : {};

Para adicionar produtos no depósito:

depositoCondominio7 [PRODUTO 137] : ADICIONAR 40;

Para consultarmos a quantidade de um produto do depósito, usamos:

CONSULTAR DEPOSITO depositoCondominio7[137];

5. CARRINHO

Assim como o DEPOSITO, é uma memória usada para guardar X unidades de Y produtos, mas é mais uma memória temporária usada pelo usuário, guardando produtos escolhidos pelo usuário, onde primeiro precisa ser criado:

CRIAR_CARRINHO carrinhoJoao : [];

Para adicionar produtos no carrinho:

carrinhoJoao ADICIONAR depositoCondominio7 [137] : 7;

Para consultarmos a quantidade de um produto no carrinho ou todos os produtos no carrinho, usamos:

CONSULTAR CARRINHO carrinhoJoao[137];

CONSULTAR CARRINHO carrinhoJoao;

6. MENSAGEM

MENSAGEM ("Teste");

Instrução de print de string

7. VENDER

VENDER carrinhoJoao[137] SCANEAR 3;

Operação que vai executar a compra de um determinado produto presente no carrinho, onde a quantidade X que é scaneada subtrai a quantidade que está na lista

8. GUARDAR

GUARDAR cond = 7 * 13;

Registrador que usamos como valor auxiliar durante as operações

9. VERIFICAR/VERIFICAR_CONSTANTEMENTE

VERIFICAR 1 < 2

@@

Verificação de condição que caso seja true, vai executar as operações até chegar em @@

VERIFICAR_CONSTANTEMENTE 1 < 2

$$

Verificação de condição que caso seja true, vai executar as operações até chegar em $$

10. PAGAMENTO

PAGAMENTO PIX APROVADO/RECUSADO;

O pagamento finaliza tudo, se o pagamento foi RECUSADO, cancela as operações, caso seja APROVADO, continua para a etapa de FINALIZAR

OPERADORES/Expressões

1. Cálculo matemático/Atribuição

+

-

*

/

=

:

2. Operadores binários

ALEM_DE(And)

OU(Or)

<

>

<=

>=

==

EXEMPLOS

1. Exemplo Simples → Programa | Saída

```
INICIAR

MENSAGEM ("Teste simples");

PRODUTO presuntoSadia = 137 PRECO 4;

CRIAR_DEPOSITO depositoCondominio13 : {};
depositoCondominio13 [PRODUTO 137] : ADICIONAR 300;
CONSULTAR DEPOSITO depositoCondominio13[137];

CRIAR_CARRINHO carrinhoParnaiba : [];
carrinhoParnaiba ADICIONAR depositoCondominio13[137] : 3;
CONSULTAR CARRINHO carrinhoParnaiba[137];

PAGAMENTO CREDITO APROVADO;

FINALIZAR
```

```
========= Mercadinho do Jota no seu prédio ============

"Teste simples"
Depósito depositoCondominio13[presuntoSadia]: 300 unidades
Carrinho carrinhoParnaiba[presuntoSadia]: 3 unidades

Pagamento: CRÉDITO
Status: APROVADO

Imprimir nota fiscal(s/n)? s

=== NOTA FISCAL ===
Produtos vendidos:
- 3x presuntoSadia (carrinho carrinhoParnaiba): R$ 12.00
Total da venda: R$ 12.00
==================

Compra finalizada
```

2. Exemplo Incorreto → Programa | Saída

```
INICIAR

MENSAGEM ("Teste simples errado");

PRODUTO presuntoSadia = 137 PRECO 4;

CRIAR_DEPOSITO depositoCondominio13 : {};
depositoCondominio130000000 [PRODUTO 137] : ADICIONAR 1;
CONSULTAR DEPOSITO depositoCondominio13[137];

CRIAR_CARRINHO carrinhoParnaiba : [];
carrinhoParnaiba ADICIONAR depositoCondominio13[137] : 3;
CONSULTAR CARRINHO carrinhoParnaiba[137];

PAGAMENTO CREDITO APROVADO;

FINALIZAR
```

```
========= Mercadinho do Jota no seu prédio =============

"Teste simples errado"
[Semântico] [Semântico] Depósito 'depositoCondominio130000000' não existe
```

3. Exemplo de cálculo Matemático → Programa | Saída

```
INICIAR

MENSAGEM ("Teste Matemático");

GUARDAR x = 7;
GUARDAR y = 3;

GUARDAR z = x + y;

CONSULTAR z;

FINALIZAR
```

```
========= Mercadinho do Jota no seu prédio =============

"Teste Matemático"
z = 10.00
'VirtualMachine' object has no attribute 'pagamento_status'
```

→ Como não foi feito nenhuma compra, não se tem nenhum pagamento