ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Relatório Final
Vending Machine

PMR 3402 –Sistemas Embarcados

Alessandro Brugnera Silva 10334040

São Paulo, 29 de abril de 2020

Sumário

1 Resumo

O projeto tem como objetivo o desenvolvimento de um protótipo de uma Vending Machine utilizando conceitos de máquina de estados e documentação via UML.

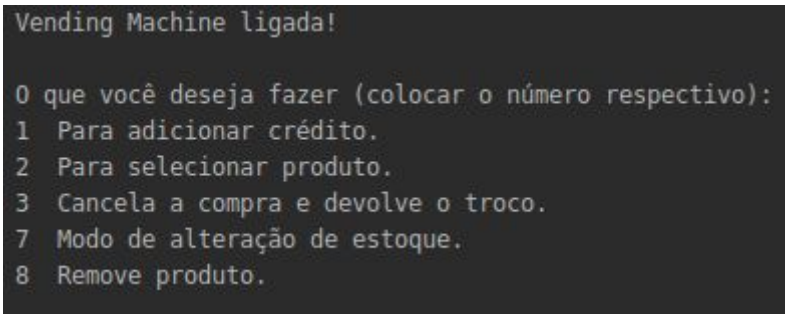
2 Documentação

A documentação devidamente atualizada para a entrega final está incluída no .zip.

3 Funcionamento

3.1 Máquina de Estados

Primeiramente ao rodar o script o usuário a Vending Machine é ligada e se encontra no estado IDLE esperando o input do usuário ou vendedor (no caso via input no terminal):

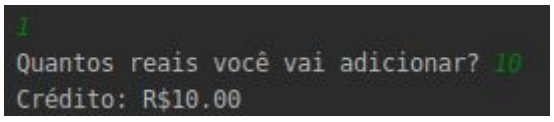


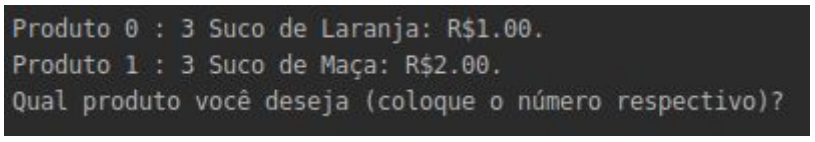
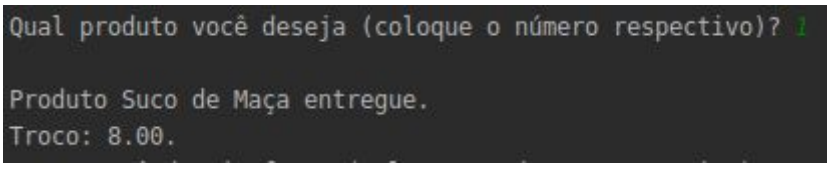
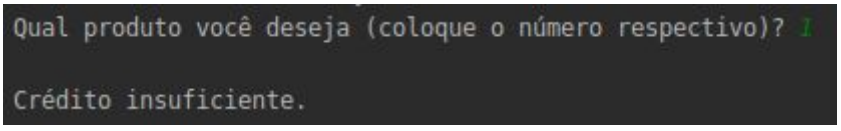
Como se trata de uma simulação as opções do usuário e vendedor estão selecionáveis na mesma tela.

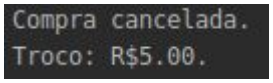
Ao realizar cada conjunto de ações (mostradas abaixo) se retorna ao IDLE - seguindo o modelo de uma máquina de estados.

3.2 Usuário

Com isso o usuário pode agir de 3 maneiras:

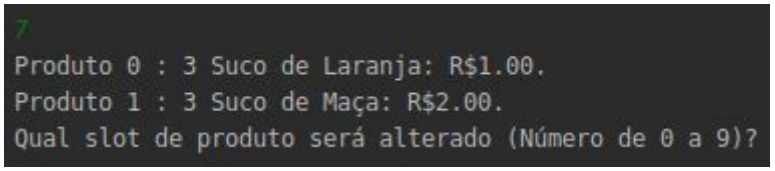
1. Inserindo crédito.
    a. 
    ```
    1
    Quantos reais você vai adicionar? 10
    Crédito: R$10.00
    ```
    b. A opção de adicionar crédito (1) é selecionado.
    c. E o valor inserido é colocado no terminal.
    d. Um feedback do crédito total é retornado.

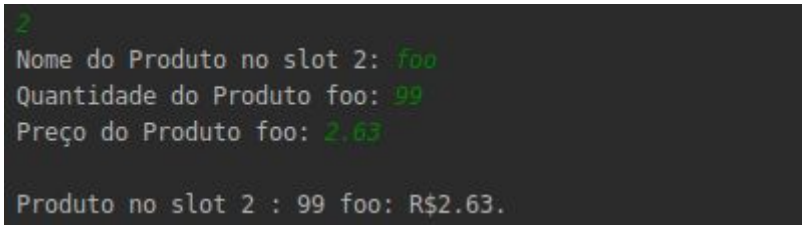
2. Selecionando produto.
    a. 
    ```
    Produto 0 : 3 Suco de Laranja: R$1.00.
    Produto 1 : 3 Suco de Maça: R$2.00.
    Qual produto você deseja (coloque o número respectivo)?
    ```
    b. A opção de adicionar compra (2) é selecionada.
    c. Primeiramente o estoque, ou seja, as opções de compra são mostradas.
    d. Segundamente o usuário seleciona o produto pelo número mostrado no estoque via terminal, com 2 possíveis opções:
        i. Crédito suficiente para compra.
            1. 
            ```
            Qual produto você deseja (coloque o número respectivo)? 1

            Produto Suco de Maça entregue.
            Troco: 8.00.
            ```
            2. Assim o produto entregado pela máquina é exposto.
            3. E o troco também é mostrado.
        ii. Crédito suficiente para compra.
            1. 
            ```
            Qual produto você deseja (coloque o número respectivo)? 1

            Crédito insuficiente.
            ```
            2. O feedback de crédito insuficiente é mostrado.

3. Cancelando a compra.
    a. 
    ```
    Compra cancelada.
    Troco: R$5.00.
    ```
    b. A opção de cancelamento (3) é selecionada.
    c. Um feedback do troco é retornado.

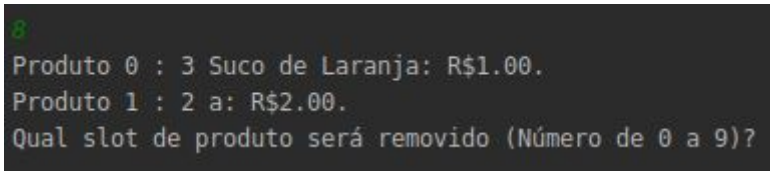
3.3 Vendedor

Simulando opções do vendedor:

7. Alterando estoque
    a. 

    b. A opção de alteração de estoque (7) é selecionada.
    c. Primeiramente o estoque atual, ou seja, as opções de compra são mostradas.
    d. Segundamente o slot a ser alterado é selecionado pelo número respectivo ao slot. Podendo ser um slot vazio.
    e. 

    f. Em seguida as informações do slot são preenchidas respectivamente:
        i. Nome.
        ii. Quantidade.
        iii. Preço
    g. Com isso uma confirmação da alteração é retornada.

8. Removendo produtos de um slot
    a. 

    b. A opção de remoção de produto do slot (8) é selecionada.
    c. Primeiramente o estoque atual, ou seja, as opções de compra são mostradas.
    d. Segundamente o slot a ser removido é selecionado pelo número respectivo ao slot.

4 Conclusão e comentário

Considero uma matéria muito útil pois ajuda a “profissionalizar” o que cada aluno sabe de programação - ainda mais com a programação em alta no mercado. A documentação em UML é algo importante e para mim foi novo e acredito que possa ser bem útil.

Além disso a parte de uma máquina de estados foi interessante para mim pois fiz um projeto que usava esse conceito, mas não conhecia a técnica. Com isso vi a utilidade da técnica formalizada. No projeto, demorei muito para esquematizar o funcionamento da máquina - com a técnica seria muito mais rápido.

A respeito da matéria acho interessante um feedback do que cada aluno fez semanalmente - ajuda bastante. Se um aluno sentir necessidade de um feedback individual seria interessante essa possibilidade também.

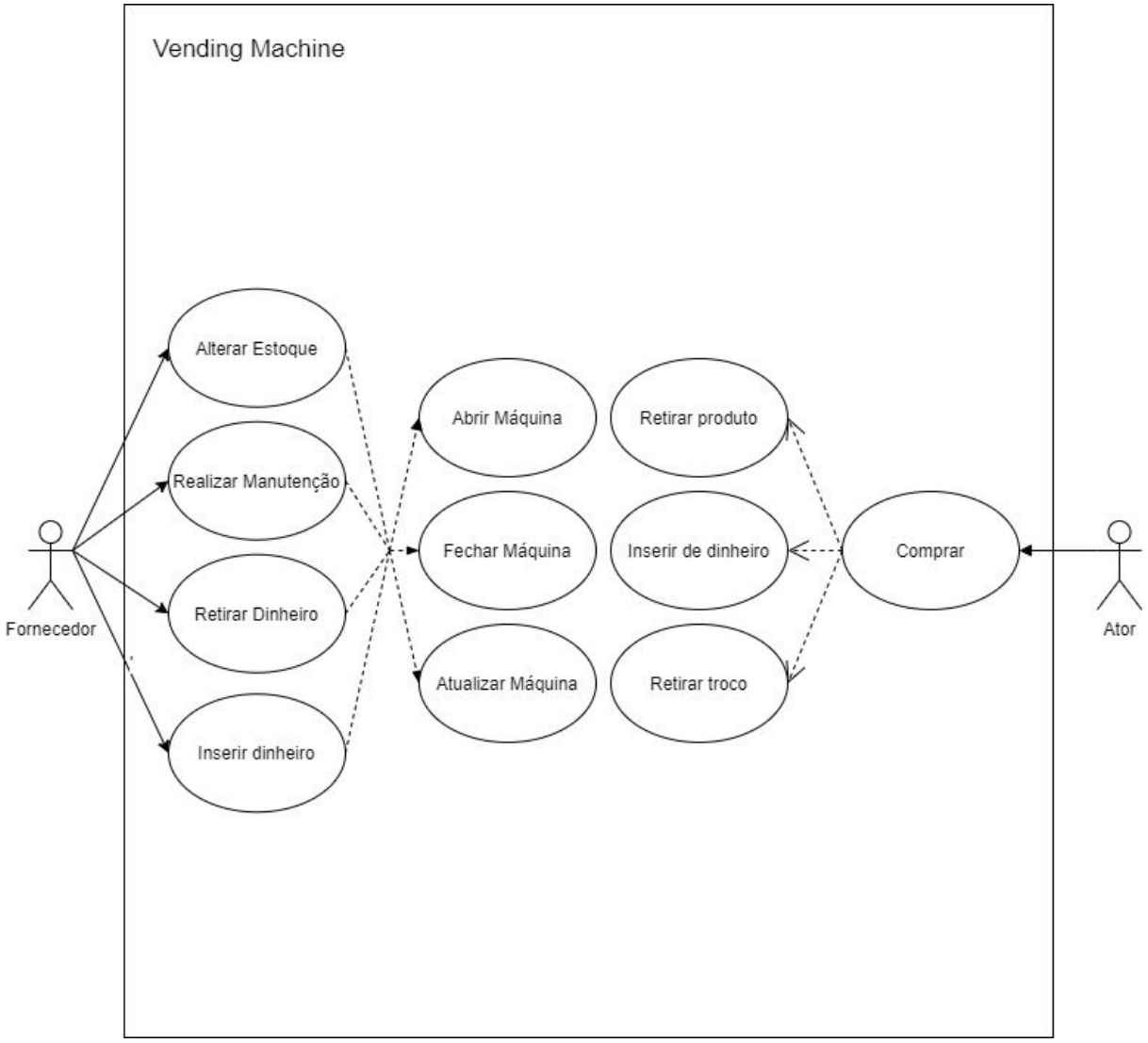
Com a pandemia, a matéria se tornou EAD. Isso ajudou muito, porque as aulas podem ser assistidas a qualquer momento - ajudando na hora correção e criação do UML e da entrega final.

A parte do C ser novo para muitos é importante, porque quem programa uma linguagem programa qualquer outra - ou seja só pesquisar. O C é mais chatinho por não ter OO - mas a apresentação da estrutura de dados facilitou o entendimento já.

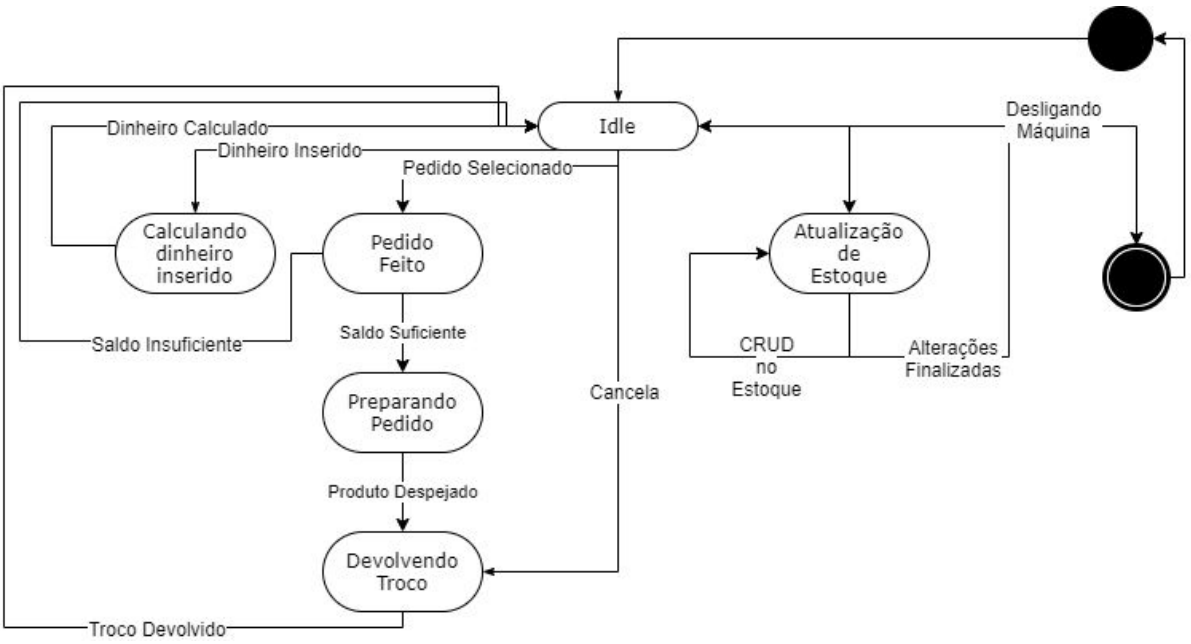
A respeito da avaliação, não consigo imaginar uma prova para a matéria - ainda mais no contexto da pandemia. Assim não vejo o porquê de uma prova. Acho que trabalhos fazem mais sentido para o aprendizado.

5 Diagramas

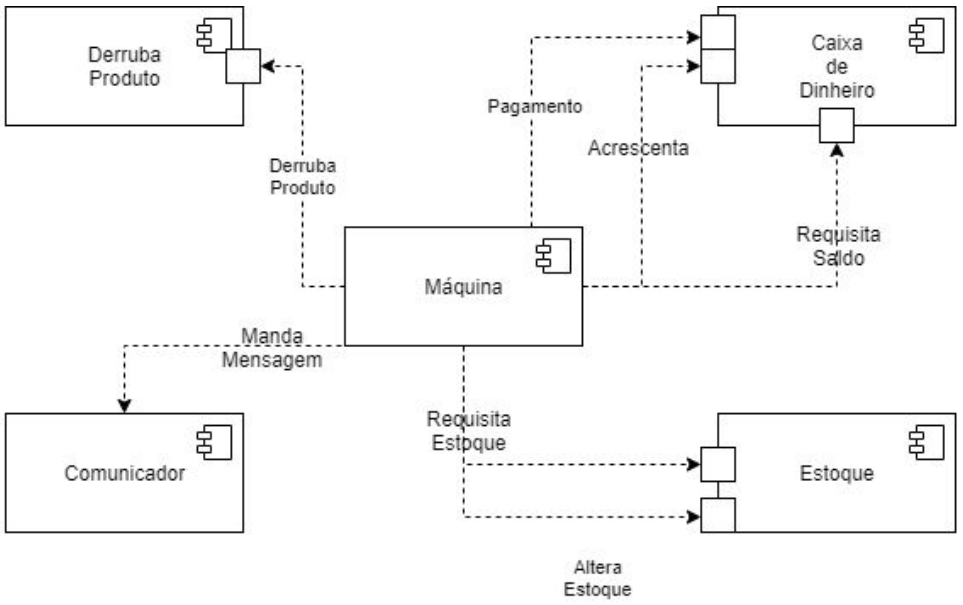
1. Casos de Uso



2. Estados


Idle
Dinheiro Calculado
Dinheiro Inserido
Pedido Selecionado
Desligando Máquina
Calculando dinheiro inserido
Pedido Feito
Atualização de Estoque
Saldo Insuficiente
Saldo Suficiente
CRUD no Estoque
Alterações Finalizadas
Preparando Pedido
Cancela
Produto Despejado
Devolvendo Troco
Troco Devolvido

3. Componentes



Requisita Estoque
Input: Void
Output: JSON

Requisita Saldo
Input: Void
Output: float

Requisita Estoque
Input: Void
Output: Bool

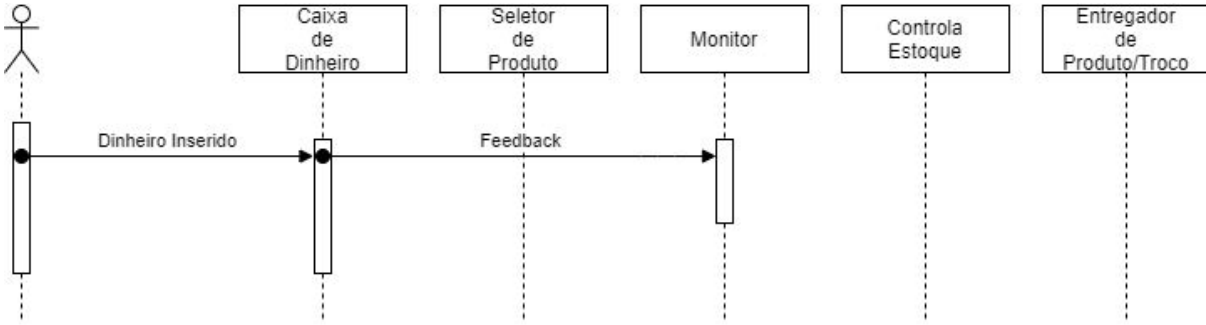
Acrescenta
Input: Float
Output: Bool

Derruba Produto
Input: int
Output: bool

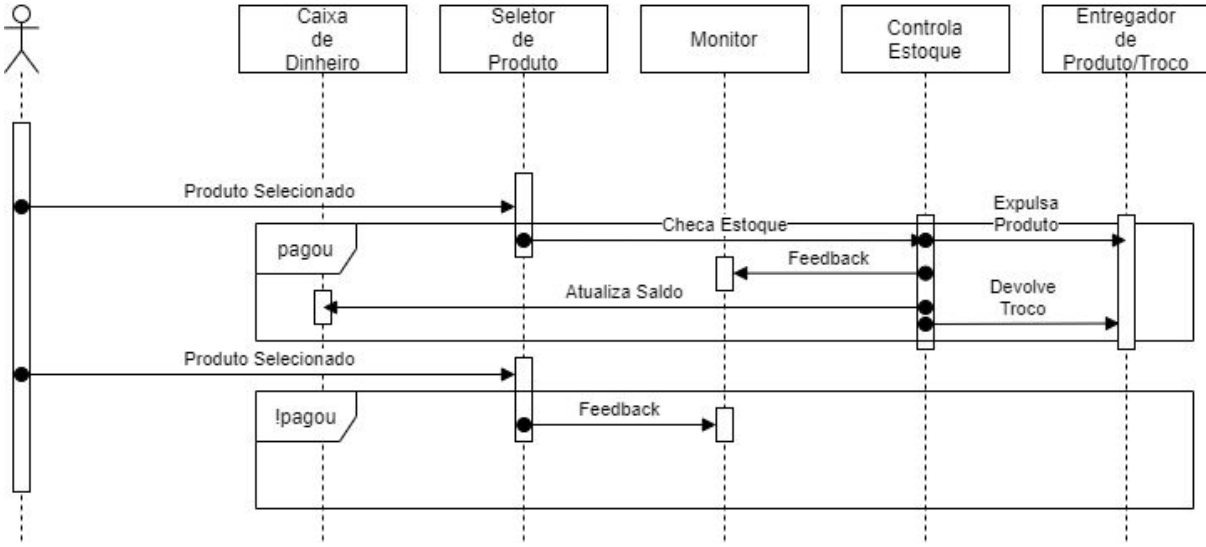
Altera Estoque
Input: int
Output: bool

4. Sequências

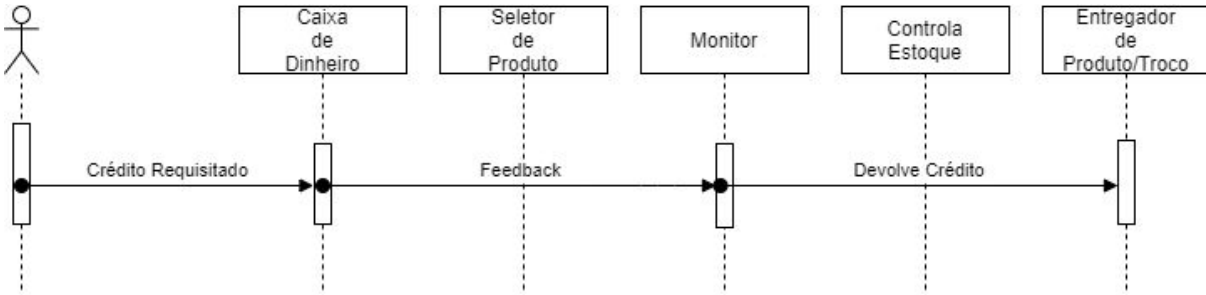
a. Inserindo crédito



b. Selecionando produto



c. Cancelando a compra



6 Códigos

main.c

```
#include <stdio.h>
#include <stdbool.h>
#include "definicoes.h"
#include "estoque.h"
#include "caixaDeDinheiro.h"
#include "comunicador.h"

int acaoMatrizTransicaoEstados[NUM_ESTADOS][NUM_EVENTOS];
int proximoEstadoMatrizTransicaoEstados[NUM_ESTADOS][NUM_EVENTOS];

void iniciaMaquinaEstados(void) {
    int i;
    int j;

    for (i = 0; i < NUM_ESTADOS; i++) {
        for (j = 0; j < NUM_EVENTOS; j++) {
            acaoMatrizTransicaoEstados[i][j] = NENHUMA_ACAO;
            proximoEstadoMatrizTransicaoEstados[i][j] = i;
        }
    }

    proximoEstadoMatrizTransicaoEstados[IDLE][INSERIRDINHEIRO] =
INSERINDO;
    acaoMatrizTransicaoEstados[IDLE][INSERIRDINHEIRO] = A01;

    proximoEstadoMatrizTransicaoEstados[INSERINDO][DINHEIROINSERIDO] =
IDLE;
    acaoMatrizTransicaoEstados[INSERINDO][DINHEIROINSERIDO] = A02;


    proximoEstadoMatrizTransicaoEstados[IDLE][PEDIR] = PEDINDO;
    acaoMatrizTransicaoEstados[IDLE][PEDIR] = A03;

    proximoEstadoMatrizTransicaoEstados[PEDINDO][PEDIDOSUCESSO] = IDLE;
    acaoMatrizTransicaoEstados[PEDINDO][PEDIDOSUCESSO] = A04;


    proximoEstadoMatrizTransicaoEstados[IDLE][CANCELAR] = CANCELANDO;
    acaoMatrizTransicaoEstados[IDLE][CANCELAR] = A09;
```

```
    proximoEstadoMatrizTransicaoEstados[IDLE][ALTERARESTOQUE] =
ALTERANDOESTOQUE;
    acaoMatrizTransicaoEstados[IDLE][ALTERARESTOQUE] = A05;


proximoEstadoMatrizTransicaoEstados[ALTERANDOESTOQUE][ESTOQUEALTERADO] =
IDLE;
    acaoMatrizTransicaoEstados[ALTERANDOESTOQUE][ESTOQUEALTERADO] = A06;


    proximoEstadoMatrizTransicaoEstados[IDLE][REMOVERPRODUTO] =
REMOVENDOPRODUTO;
    acaoMatrizTransicaoEstados[IDLE][REMOVERPRODUTO] = A07;


proximoEstadoMatrizTransicaoEstados[REMOVENDOPRODUTO][PRODUTOREMOVIDO] =
IDLE;
    acaoMatrizTransicaoEstados[REMOVENDOPRODUTO][PRODUTOREMOVIDO] = A08;


}

void iniciaSistema(void) {
    iniciaMaquinaEstados();
    est_inicializaEstoque();
    cxd_zeraCaixa();
    printf("Vending Machine ligada!\n");
}


int executaAcao(int codigoAcao) {
    int eventoInterno = NENHUM_EVENTO;

    if (codigoAcao == NENHUMA_ACAO) {
        return NENHUM_EVENTO;
    }

    switch (codigoAcao) {
        break;
        case A01:
            com_inserindoDinheiro();
            eventoInterno = DINHEIROINSERIDO;
            break;
```

```
        case A02:
            com_dinheiroInserido();
            break;

        case A03:
            com_pedindo();
            eventoInterno = PEDIDOSUCESSO;
            break;

        case A04:
            com_pedidoRecebido();
            break;

        case A05:
            com_alterandoEstoque();
            eventoInterno = ESTOQUEALTERADO;
            break;

        case A06:
            com_estoqueAlterado();
            break;

        case A07:
            com_removendoProduto();
            eventoInterno = PRODUTOREMOVIDO;
            break;

        case A08:
            com_produtoRemovido();
            break;

        case A09:
            com_cancelamento();
            break;

        default:
            printf("Ação não identificada.");
            break;
    }

    return eventoInterno;
}
```

```
int obterEvento() {
    //TODO: recebe input
    int teclaRecebida = com_inputSeletor();

    switch (teclaRecebida) {
        case PEDINDO:
            return PEDINDO;
            break;
        case INSERINDO:
            return INSERINDO;
            break;
        case CANCELANDO:
            return CANCELANDO;
            break;
        case ALTERANDOESTOQUE:
            return ALTERANDOESTOQUE;
            break;
        case REMOVENDOPRODUTO:
            return REMOVENDOPRODUTO;
            break;
        default:
            printf("Comando desconhecido."); //TODO:avaliar se eh
gambiarra
            break;
    }



    return NENHUM_EVENTO;
}

int obterProximoEstado(int estado, int evento) {
    return proximoEstadoMatrizTransicaoEstados[estado][evento];
}

int obterAcao(int estado, int evento) {
    return acaoMatrizTransicaoEstados[estado][evento];
}
```

```
int main(void) {

    int codigoEvento;
    int codigoAcao;
    int estado;
    int eventoInterno;

    estado = IDLE;
    eventoInterno = NENHUM_EVENTO;

    iniciaSistema();

//    loop de maquina de estados:
    while (true) {
        if (eventoInterno == NENHUM_EVENTO) {
            codigoEvento = obterEvento();
        } else {
            codigoEvento = eventoInterno;
        }
        if (codigoEvento != NENHUM_EVENTO) {
            codigoAcao = obterAcao(estado, codigoEvento);
            estado = obterProximoEstado(estado, codigoEvento);
            eventoInterno = executaAcao(codigoAcao);
//            printf("\nEstado: %d Evento: %d Acao: %d\n", estado,
codigoEvento, codigoAcao);


        }
    }

}
```

definicoes.h

```
#ifndef VENDINGMACHINE_DEFINICOES_H
#define VENDINGMACHINE_DEFINICOES_H

#define NUM_ESTADOS 5
#define NUM_EVENTOS 10

#define NENHUM_EVENTO -1
#define INSERIRDINHEIRO 1
#define DINHEIROINSERIDO 4
```

```
#define PEDIR 2
#define PEDIDOSUCESSO 5
#define CANCELAR 3
#define ALTERARESTOQUE 7
#define ESTOQUEALTERADO 9
#define REMOVERPRODUTO 8
#define PRODUTOREMOVIDO 10


#define NENHUMA_ACAO -1
#define A01 1
#define A02 2
#define A03 3
#define A04 4
#define A05 5
#define A06 6
#define A07 7
#define A08 8
#define A09 9


#define IDLE 0
#define PEDINDO 1
#define INSERINDO 2
#define CANCELANDO 3
#define ALTERANDOESTOQUE 4
#define REMOVENDOPRODUTO 5

#endif //VENDINGMACHINE_DEFINICOES_H
```

estoque.h

```
#ifndef VENDINGMACHINE_ESTOQUE_H
#define VENDINGMACHINE_ESTOQUE_H

#define NUMS_SLOTSPRODUTOS 10


extern void est_inicializaEstoque();

extern void est_exportarEstoque();

extern int est_pagamento(int);
```

```
extern char* est_getNomeProduto(int);
extern float est_getPrecoProduto(int);

extern void est_alteraEstoque(int);

extern int est_removeEstoque(int);

#endif //VENDINGMACHINE_ESTOQUE_H
```

estoque.c

```
#include <stdio.h>
#include <stdbool.h>
#include "estoque.h"
#include "string.h"
#include "caixaDeDinheiro.h"


struct produto {
    char nome[30];
    float preco;
    int quantidade;
};

static struct produto produtos[NUMS_SLOTSPRODUTOS];


void est_inicializaEstoque(void) {
    for (int i = 0; i < NUMS_SLOTSPRODUTOS; i++) { //ver se existe no
estoque
        produtos[i].quantidade = -1;
        produtos[i].preco = 0.0;
    }


    //Já liga com isso
    strncpy(produtos[0].nome, "Suco de Laranja", 29);
    produtos[0].nome[29] = 0;
    produtos[0].preco = 1.00;
    produtos[0].quantidade = 3;

    strncpy(produtos[1].nome, "Suco de Maça", 29);
    produtos[1].preco = 2.00;
    produtos[1].quantidade = 3;
```

```
}

void est_exportarEstoque() { //TODO: tentar printar isso pelo
comunicador
    fflush(stdin);
    for (int i = 0; i < NUMS_SLOTSPRODUTOS; i++) {
        if (produtos[i].quantidade > -1) {
            fflush(stdin);
            printf("Produto %i : ", i+1);
            printf("%i ", produtos[i].quantidade);
            printf("%s", produtos[i].nome);
            printf(": R$%.2f.\n", produtos[i].preco);
        }
    }

}

int est_pagamento(int produtoDesejado) {
    if (cxd_getCredito() >= produtos[produtoDesejado].preco) {
        return true;
    } else {
        return false;
    }
}

char* est_getNomeProduto(int produtoDesejado) {
    return  produtos[produtoDesejado].nome;
}

float est_getPrecoProduto(int produtoDesejado) {
    return  produtos[produtoDesejado].preco;
}

void est_alteraEstoque(int slotAlterado) {
    printf("Nome do Produto no slot %i: ", slotAlterado);
    fflush(stdin);
    scanf("%s", produtos[slotAlterado].nome);
    printf("Quantidade do Produto %s: ", produtos[slotAlterado].nome);
    fflush(stdin);
    scanf("%d", &produtos[slotAlterado].quantidade);
    printf("Preço do Produto %s: ", produtos[slotAlterado].nome);
    fflush(stdin);
    scanf("%f", &produtos[slotAlterado].preco);
    printf("\n");
```

```
    printf("Produto no slot %i : ", slotAlterado);
    printf("%i ", produtos[slotAlterado].quantidade);
    printf("%s", produtos[slotAlterado].nome);
    printf(": R$%.2f.\n", produtos[slotAlterado].preco);
    printf("\n");
}

int est_removeEstoque(int slotAlterado) {
    strncpy(produtos[slotAlterado].nome, "", 29);
    produtos[slotAlterado].nome[29] = 0;
    produtos[slotAlterado].quantidade = -1;
    produtos[slotAlterado].preco = 0.0;

    return true;
}
```

caixaDeDinheiro.h

```
#ifndef VENDINGMACHINE_CAIXADEDINHEIRO_H
#define VENDINGMACHINE_CAIXADEDINHEIRO_H
extern void cxd_zeraCaixa();

extern float cxd_getCredito();

extern void cxd_adicionar(float);

#endif //VENDINGMACHINE_CAIXADEDINHEIRO_H
```

caixaDeDinheiro.c

```
#include "caixaDeDinheiro.h"

static float credito;

void cxd_zeraCaixa() {
    credito = 0;
}

float cxd_getCredito() {
    return credito;
}

void cxd_adicionar(float dinheiroInserido) {
    credito += dinheiroInserido;
```

```
}
```

comunicador.h

```h
#ifndef VENDINGMACHINE_COMUNICADOR_H
#define VENDINGMACHINE_COMUNICADOR_H

extern int com_inputSeletor();

extern void com_inserindoDinheiro();
extern void com_dinheiroInserido();

extern void com_pedindo();
extern void com_pedidoRecebido();

extern void com_alterandoEstoque();
extern void com_estoqueAlterado();

extern void com_removendoProduto();
extern void com_produtoRemovido();

extern void com_cancelamento();
#endif //VENDINGMACHINE_COMUNICADOR_H
```

comunicador.c

```c
#include <stdio.h>
#include "comunicador.h"
#include "caixaDeDinheiro.h"
#include "string.h"
#include "estoque.h"


int com_inputSeletor() {
    int evento;
    printf("\n");
    printf("O  Que você deseja fazer (colocar o número respectivo):");
    printf("\n");
    printf("1  Para adicionar crédito.");
    printf("\n");
    printf("2  Para selecionar produto.");
    printf("\n");
    printf("3  Cancela a compra e devolve o troco.");
```

```
    printf("\n");
    printf("7  Modo de alteração de estoque.");
    printf("\n");
    printf("8  Remove produto.");
    printf("\n");
    printf("9  Desliga a máquina.");
    fflush(stdin);
    printf("\n");
    scanf("%d", &evento);

    return evento;
}

void com_inserindoDinheiro() {
    printf("Quantos reais você vai adicionar? ");
}

void com_dinheiroInserido() {//TODO: Try Catch
    float dinheiroInserido;
    fflush(stdin);
    scanf("%f", &dinheiroInserido);
    cxd_adicionar(dinheiroInserido);
    fflush(stdin);
    printf("Crédito: R$%.2f", cxd_getCredito());
}

void com_pedindo(void) {
    est_exportarEstoque();
    fflush(stdin);
    printf("Qual produto você deseja (coloque o número respectivo)? ");
}

void com_pedidoRecebido() {//TODO: Try Catch
    fflush(stdin);
    int produtoDesejado;
    scanf("%d", &produtoDesejado);

    if (est_pagamento(produtoDesejado)) {
        printf("\n");
        printf("Produto %s entregue.",
               est_getNomeProduto(produtoDesejado)); //TODO: conferir se
isso com ponteiro vai dar bom
        printf("\n");
        printf("Troco: %.2f.", cxd_getCredito() -
est_getPrecoProduto(produtoDesejado));
```

```
    } else {
        printf("\n");
        printf("Crédito insuficiente.");
    }
}

void com_alterandoEstoque() {
    fflush(stdin);
    est_exportarEstoque();
    fflush(stdin);
    printf("Qual slot de produto será removido (Número de 1 a 10)?\n");
}

void com_estoqueAlterado() {
    fflush(stdin);
    int slotAlterado;
    scanf("%d", &slotAlterado);

    est_alteraEstoque(slotAlterado);
}

void com_removendoProduto() {
    fflush(stdin);
    est_exportarEstoque();
    fflush(stdin);
    printf("Qual slot de produto será removido (Número de 1 a 10)?\n");
}

void com_produtoRemovido() {
    fflush(stdin);
    int slotAlterado;
    scanf("%d", &slotAlterado);


    if (est_removeEstoque(slotAlterado)) {
        printf("Slot %i vazio. ", slotAlterado);
    } else {
        printf("Slot inexistente.");
    }

}

void com_cancelamento() {
    printf("Compra cancelada.\n");
    printf("Troco: R$%.2f.\n", cxd_getCredito());
```

```
    cxd_zeraCaixa();
}
```