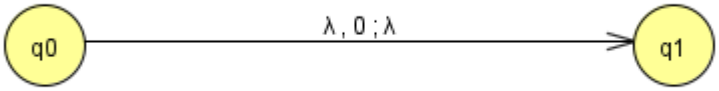
Simulador de AP, Representação gráfica

Q e conjunto finito de estados = q0 q1, q2.

linguagem de entrada L = { w ∈ {0,1}*}.

primeiro irá pedir o PUSH.

depois o pop



Depois ira executar esse passo ate esvaziar o Automato completamente.

linguagem de entrada = {0,1}.

https://github.com/SirSaito/APP.git

```
/*
Descrição do Trabalho de Automato e Pilha:

"Desenvolver um AAP (Autômato a Pilha), ele deve simular o
funcionamento teórico de um autômato.

Requisitos mínimos:

Uma cadeia de entrada (definida pelo alfabeto desejado) e dizer se
reconhece ou não
Definir em código toda a sêxtupla (mesmo que não utilize todas as
variáveis)
Ideias de extras:

Fazer a definição dos estados dinâmicos
Definir a linguagem
Definir alfabeto, dentre outros elementos da sêxtupla como entrada.
BONUS: Fazer representação gráfica"
*/

#include<stdio.h>   // biblioteca stdio.
```

```c
#include<stdlib.h>  // biblioteca stdlib.
#include<string.h>
#include<locale.h>  // biblioteca locale.

typedef struct NO{  // Estrutura do interior da Pilha.
    char dado[100];   //
    struct NO *prox;    //
} NO;

typedef struct PILHA{ //Topo da Pilha.
    NO *topo;   // Tamanho da pilha.
} PILHA;

void inicializaPilha(PILHA *p){ //Topo aponta para o primeira
referencia da Pilha.
    p -> topo = NULL;
}

void empilha(char dado[], PILHA *p){ //cria espaco na Pilha e coloca o
primeiro elemento.
    NO *ptr = (NO*) malloc(sizeof(NO));

    if(ptr == NULL){
        printf("/t/tErro de alocacao de novo no./n");
        return;
    }

    else{
            strcpy(ptr->dado, dado);     //ptr o espaco da pilha aponta
para dado.
            ptr -> prox = p -> topo;    //prox do elemnto aponta pata o
topo.
            p -> topo = ptr;    //elemento enserido no topo.
        }
}

char *desempilha(PILHA *p){ //Aponta para dado para remover elemento.
    NO* ptr = p -> topo; //ptr apontando para o elemento no topo da
Pilha.
    char dado[100];

    if(ptr == NULL){ //Checagem se a Pilha esta vazia.
        printf("\t\tPilha vazia.\n");
```

```
        return 0;
    }

    else{ //Desempilha Elemento.
        p -> topo = ptr -> prox;
        ptr -> prox = NULL;
        strcpy(dado, ptr->dado);
        free (ptr);
        return dado;
    }
}

void imprimePilha(PILHA *p){ //Imprime a Pilha.
    NO *ptr = p -> topo;

    if(ptr == NULL){ //Checagem se a Pilha esta vazia.
        printf("\t\tPilha vazia.\n");
        return;
    }

    else{
        while (ptr != NULL){
            printf("\t\t%s\n", ptr -> dado);
            ptr = ptr -> prox;
        }
    }
}

int main(){
    PILHA *p1 = (PILHA*) malloc(sizeof(PILHA)); //Chama Pilha no
Main().
    int i, x, y, o, r, m=0;
    char dado[100];
    char target_char = '0';
    char target_char1 = '1';
    int k = 0, j = 0;

    printf("\t\tBem vindo ao simulador de Automato a Pilha.\n");
//Mensagem de Bem vindo.
    printf("\t\tPara fins praticos o alfabeto que sera usado sera
binario (0/1)\n");   //Mensagem de definicao de linguagem do Automato.
    printf("\t\tUm Automato a Pilha e definido por uma sextupla:\n");
//Mensagem de definicao do estado inicial.
```

```
    printf("\t\t1 é um conjunto finito de estados = q0 q1, q2.\n");
    printf("\t\t2 é um conjunto finito de símbolos, denominado alfabeto
de entrada = {0, 1}\n");
    printf("\t\t3 é um conjunto finito de símbolos, denominado alfabeto
da pilha = {A, Z}");
    printf("\t\t4 é a relação de transição.\n");
    printf("\t\t5 é o estado inicial = q0\n");
    printf("\t\t6 é o conjunto de estado(s) finai(s) (ou de aceitação)
= q2\n");
    printf("\t\tA seguir sera pedido para enserir elementos na Pilha
com a linguagem L = { w ∈ {0,1}*}\n");    //Mensagem de definicao dos
ESTADOS DE TRANSICAO.
    printf("\t\tIsso quer dizer que a linguagem so aceita o mesmo
numero de 0's e 1's, sem se importar com a ordem.\n");

    if (p1 == NULL){
        printf("/t/tErro de alocacao da Pilha./n");
        exit(0);
    }

    else{

        inicializaPilha(p1);
                    printf("\t\tDeseja comecar?(Digite 1 para sim e 0
para nao)\n");
                        scanf("%d", &x);
                    while(x != 0){
                            printf("\t\tDigite um elemento para alocar
na pilha(PUSH)\n");
                            int c;
                             while ((c = getchar()) != '\n' && c !=
EOF);

                            fgets(dado, sizeof(dado), stdin);
                            dado[strcspn(dado, "\n")] = 0;
                             for (i = 0; dado[i] != '\0'; i++) {
                                if (dado[i] == target_char) {
                                    k++;
                                }
                                if (dado[i] == target_char1) {
                                    j++;
                                }
                            }
```

```
                            if((k == j)&&(k>0)&&(j>0)){
                                printf("\t\tPilha aceita\n");
                                empilha (dado, p1);
                                m++;
                            }
                            else{
                                printf("\t\tPilha reijata\n");
                            }

                            j=0;
                            k=0;
                        printf("\t\tDeseja continuar?(Digite 1 para sim
e 0 para nao)\n");
                        scanf("%d", &x);
                        imprimePilha(p1);
                    }

        if(m > 0){
            printf("\t\tElementos da pilha:\n");
            imprimePilha(p1);
            for(i=0 ; i <= m ; i++){
                printf("\t\tTentando desempilhar(POP) - resultado:
%d\n", desempilha(p1));
                imprimePilha(p1);
            }
        }

        printf("\t\tAutomato vazio, tenha um bom dia!!!\n");
    }

    return 0;

}
```