Utilizando PILHA em uma calculadora pós-fixada: um exemplo prático com a notação Polonesa Inversa (RPN).

José Hyago Carlos de Medeiros[1]

1. INTRODUÇÃO

O presente trabalho apresenta uma abordagem sintetizada da estrutura de dados pilha, com ênfase na aplicação em expressões aritméticas com a notação Polonesa Inversa (RPN).

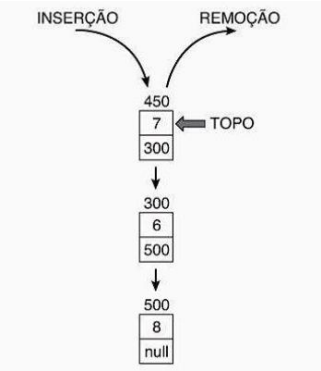
Na Estrutura de dados, existem conjuntos dinâmicos simples que usam ponteiros como os tipos de estruturas de dados (TAD) fila e a pilha que são consideradas especializadas por possuírem características próprias.

2. PILHAS (*Stacks*)

As pilhas são uma estrutura de dados muito empregadas na construção de algoritmos e de programa gerais. É um tipo abstrato de dados de armazenamento onde o último elemento que entra é o primeiro que sai (princípio LIFO – *Last in, First Out*), ou seja, "o último a entrar é o primeiro a sair", sendo assim, o último item inserido na pilha é sempre o primeiro a ser removido ou lido. Todas as inserções, retiradas e, geralmente, todos os acessos são feitos em apenas um extremo da lista (topo) e assim são colocadas um sobre o outro, o item inserido mais recente está no topo e o inserido menos recente no fundo.

O modo de funcionamento de pilha é semelhante um empilhamento de objetos, um exemplo intuitivo, é uma pilha de pratos em uma prateleira, sendo conveniente retirar ou adicionar pratos na parte superior. A figura 1 ilustra a estrutura de dados do tipo pilha.

Figura 1. Estrutura da dados do tipo pilha.



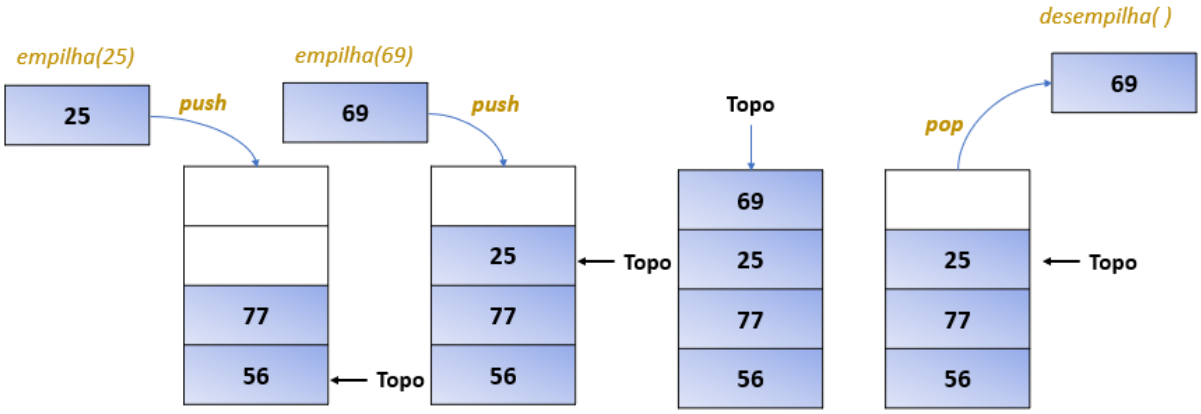
Fonte: (ASCENCIO) 2010

[1] Aluno de Mestrado em Sistemas e Computação/PPgSC na Universidade Federal do Rio Grande do Norte – Natal. E-mail: hiagocarlosm@gmail.com.

2.1. Funcionamento das pilhas

A operação de inserção em uma pilha é frequentemente denominada *PUSH* (empilha), já a remoção de um elemento do topo da pilha é a operação denominada *POP* (desempilha). A figura 2 a seguir ilustra uma pilha contendo valores numéricos e o seu topo.

Figura 2. Ilustração do funcionamento de uma pilha.



Fonte: Criada pelo autor.

2.2. Implementação do tipo pilha

Existem várias implementações possíveis do tipo pilha. Podem deferir da natureza dos elementos, maneira como são armazenados e operações disponíveis.

Podemos implementar uma pilha de no máximo *n* elementos conforme o código abaixo.

```
Pilha_vazia

se topo = 0
        retorne (verdadeiro)
senão retorne (falso)

Empilha(x,P)     - push

se topo = limite
        retorne (“erro! acima do limite”)
topo ← topo + 1
P[topo] ← x

Desempilha(P)     - pop

se Pilha_vazia
        retorne (“erro! Pilha vazia”)
senão topo ← topo – 1
        retorne (P[topo+1])
```

O código mostra os efeitos das operações modificadoras PUSH (empilhar) e POP (desempilhar). Cada uma das três operações em pilha demora o tempo $\theta(1)$.

2.3. Aplicações das pilhas

As pilhas encontram inúmeras aplicações em programação e desenvolvimento de algoritmos, como por exemplo:

- Avaliação de Expressões e Parsing Sintático.
- Gerenciamento de memória em tempo de compilação
- Operações como desfazer e refazer em aplicações
- Controle de navegação de browsers
- Analise das expressões aritméticas

3. NOTAÇÃO POLANESA INVERSA (RPN – *Reverse Polish Notation*)

A Notação Polonesa Inversa (ou RPN na sigla em inglês, de Reverse Polish Notation), também conhecida como notação pós-fixada, foi inventada pelo cientista da computação Charles Hamblin apresentando seu trabalho em meados dos anos 1950. Apesar de parecer algo estranho, tudo não passa de apenas a primeira impressão de quem não tem familiaridade com a notação, por tanto, trás vantagens como reduzir o número de passos lógicos minimiza erros de computação e maximiza a velocidade operacional.

Na tabela 1 a seguir é denotado alguns exemplos de operações e notações, por meio de contagem de números de passos lógicos operacionais para o modo RPN comparado com o modo convencional.

Operação	Notação convencional	Notação Polonesa Inversa
$a + b$	$a + b$	$a\ b +$
$\frac{a+b}{c}$	$(a + b)/c$	$a\ b + c\ /$
$a\ x\ b - \frac{c}{d}$	$((a * b) - (c/d))$	$a\ \ b *\ \ c\ \ d\ / -$

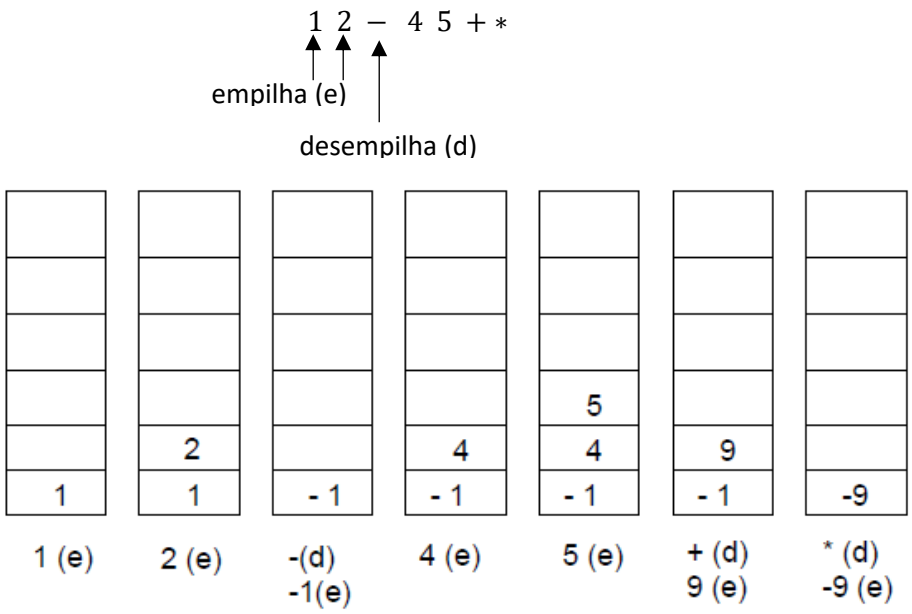
Tabela 1. Exemplos de notações e operações.

4. IMPLEMENTAÇÃO DA CALCULADORA PÓS-FIXADA

4.1 Uso em máquinas de pilhas

A notação polonesa inversa é utilizada em linguagens de programação de pilhas de dados. Um exemplo prático de aplicação em pilha é o funcionamento de calculadoras HP (Hewlett-Packard). Elas trabalham as expressões pós-fixadas, então para avaliarmos uma

expressão como $(1 - 2) * (4 + 5)$ Esta expressão pode ser representada na notação polonesa inversa como: $1\ 2 - 4\ 5 + *$. Podemos calcular numa máquina de pilha como:



Cada operando é empilhado numa pilha de valores. Ao se encontrar um operador, desempilha-se o número apropriado dos operandos (2 para operadores binários e um para valores unários), realiza a operação e empilha-se o resultado, este processo segue até o final da expressão.

4.2 Implementando a calculadora pós-fixada usando Pilha

A implementação do código foi feita na linguagem de programação C, utilizando de bibliotecas e a estrutura de dados do tipo pilha.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define MAX 100
4
5
6  int  *p;
7  int *tos;
8  int *bos;
9  void push (int i);
10 int pop (void);
11
```

```c
int main()
{
    int a, b;
    char s[80];
    p = (int *) malloc(MAX*sizeof (int)); /* Aloca memoria para a pilha */
    if(!p){
        printf("Erro de alocação de memoria\n");
        exit(1);
     }
     tos = p;
     bos = p+MAX-1;
     printf("Calculadora\n");
     do {
         printf(":");
         gets(s);
         switch(*s) {
         case '+':
             a = pop();
             b = pop();
             printf("%d\n", b+a);
             push(b+a);
             break;
         case '-':
             a = pop();
             b = pop();
             printf("%d\n", b-a);
             push(b-a);
             break;
         case '*':
             a = pop();
             b = pop();
             printf("%d\n", b*a);
             break;
        case '/':
            a = pop();
            b = pop();
            if (a==0){
                printf("divisão por 0\n");
                break;
            }
            printf("%d\n", b/a);
            push(b/a);
            break;
```

```
55          case '.':
56              a = pop();
57              push(a);
58              printf("Valor corrente no topo da pilha: %d\n",a);
59              break;
60          default:
61              push(atoi (s));
62          }
63      } while (*s!='q');
64  return 0;
65  }
66
67  /* Armazena um elemento na Pilha = empilha() */
68  void push (int i)
69  {
70      if (p>bos) {
71          printf("Pilha cheia\n");
72          return;
73      }
74      *p = i;
75      p++;
76  }
77  /* Recupera um elemento da Pilha = desempilha() */
78  pop (void)
79  {
80      p--;
81      if (p<tos){
82          printf("Pilha Vazia\n");
83          return 0;
84      }
85      return *p;
86  }
```

4.3 Complexidade do algoritmo

Considerando que o algoritmo terá que percorrer toda a expressão informada pelo usuário: $1\ 2\ -\ 4\ 5\ +*$, ou seja os n elementos nela contidos, sua complexidade é $\boldsymbol{\theta}(n)$.

5. CONCLUSÃO

A notação RPN tem larga utilização no mundo científico pela fama de permitir uma linha de raciocínio mais direta durante a formulação como também não é necessário parênteses ao utilizar a notação polonesa inversa sendo assim facilita a não ter ambiguidades sendo mais fácil (avaliar a expressão) de calcular o resultado através de uma máquina de pilha.

REFERÊNCIAS

Ascencio, Ana Fernanda Gomes. Estrutura de dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++ / Ana Fernanda Gomes Ascencio, Graziela Santos de Araújo. – São Paulo: Pearson Prentice Hall, 2010.

Algoritmos / Thomas H. Cormen... [et al.] ; [tradução Arlete Simille Marques]. - Rio de Janeiro: Elsevier, 2012. il. Tradução de: Introduction to algorithms, 3rd ed.

LinkFang. Notação polonesa inversa. Rede Paraíba de Notícias (ed.). João Pessoa, 1 jun. 2020. Disponível em: https://pt.linkfang.org/wiki/RPN. Acesso em: 3 out. 2020.

Szwarcfiter, Jayme Luiz. Estruturas de dados e seus algoritmos / Jayme Luiz Szwarcfiter, Lilian Markenzon. - 3.ed. [Reimpr.]. - Rio de Janeiro : LTC, 2015.

Referência: Notas de aulas da professora Nina Edelwais. Estrutura de dados – séries de livros didáticos – Informática – UFRGS.