Pilhas

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

Baseado nos slides do Prof. Rudinei Goularte

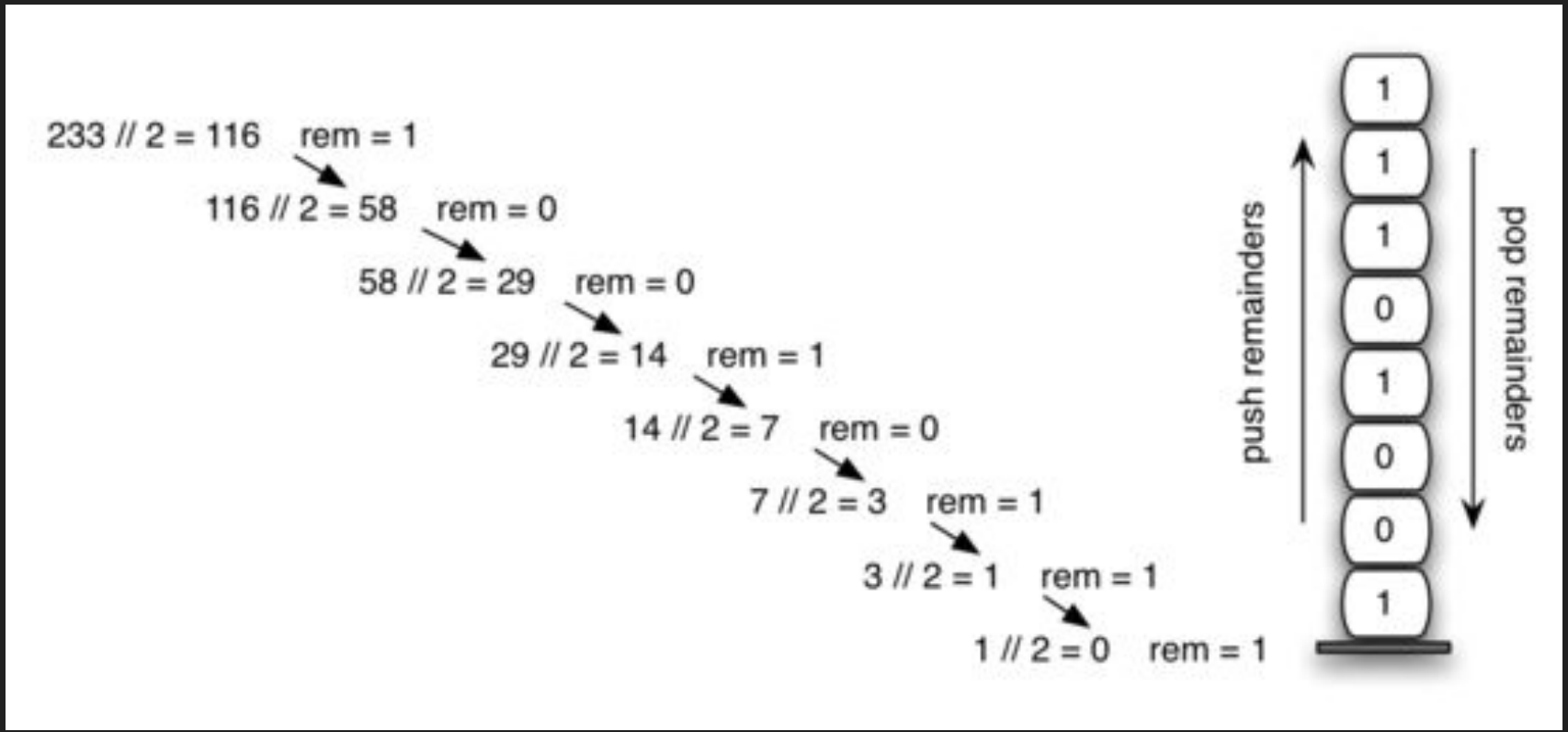
Pilhas

Conteúdo

- ➔ Conceitos Introdutórios
- ➔ Implementação Sequencial
- ➔ Implementação Encadeada
- ➔ Aplicações com Pilha

Exemplo prático

- Faça um algoritmo para converter um número decimal em sua respectiva representação binária.



Fonte:
https://panda.ime.usp.br/panda/static/pythonds_pt/03-EDBasicos/08-DecimalParaBinario.html

O que são Pilhas?

O quê são?

Exemplos do mundo real

Pra que servem as Pilhas?

Para que servem?

- Auxiliam em problemas práticos em computação
  - O botão “back” de um navegador web ou a opção “undo” de um editor de textos
  - Controle de chamada de procedimentos
  - Estrutura de dados auxiliar em alguns algoritmos como a busca em profundidade

Pilhas

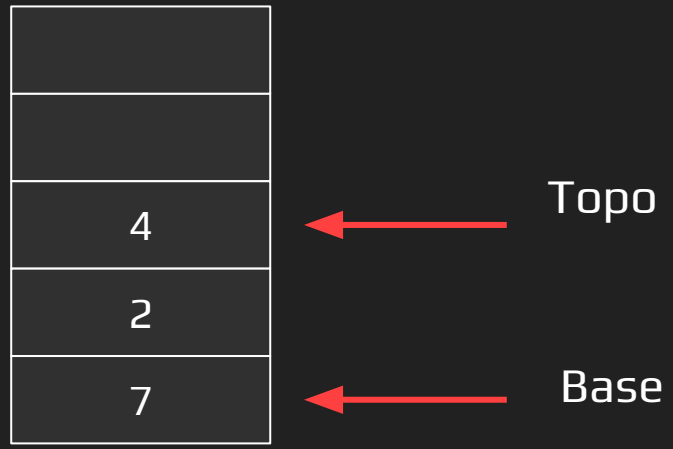
- Também conhecida como *stack*
- Estrutura de dados nas quais o último elemento inserido é o primeiro a ser retirado
  - Só é permitido acesso a 1 único item por vez
    - O último que foi inserido

Pilhas

- Pilha é construída a partir da base até o topo
  - Topo = nome dado ao último elemento inserido
- Qualquer operação com a pilha ocorre no topo

Pilhas

- Por seu tipo de acesso, é uma estrutura LIFO
  - Last-In First-Out



TAD Pilhas

- Operações principais
  - empilhar(P,x): insere o elemento x no topo de P
    - Também conhecido como *push*
  - desempilhar(P): remove o elemento do topo de P, e retorna esse elemento
    - Também conhecido como *pop*

Pilhas

- *Push*
  - Insere um elemento no topo da pilha
  - Aumenta o tamanho da pilha
  - Atualiza qual elemento é o topo

Pilhas

- *Pop*
  - Remove o elemento no topo da pilha
  - Reduz o tamanho da pilha
  - Atualiza qual elemento é o topo

Pilhas

- Duas maneiras principais de implementar uma pilha
  - Vetores
    - Geralmente, por alocação estática: reserva de memória em tempo de compilação
  - Lista Ligada
    - Por alocação dinâmica: em tempo de execução
- Mas existem variações intermediárias!

Métodos de implementação de Pilha

Organização vs. alocação de memória

1. Sequencial e estática: Uso de arrays
2. Encadeada e estática: Array simulando Mem. Princ.
3. Sequencial e dinâmica: Alocação dinâmica de Array
4. Encadeada e dinâmica: Uso de ponteiros

Alocação da memória \ Organização da memória:	Sequencial	Encadeada
Estática	1	2
Dinâmica	3	4

Pilhas

- Vamos ver um exemplo!
- https://www.cs.usfca.edu/~galles/visualization/StackArray.html

Implementando um TAD de Pilha

TAD Pilhas

- Operações auxiliares
  - criar(P): cria uma pilha P vazia
  - apagar(P): apaga a pilha P da memória
  - topo(P): retorna o elemento do topo de P, sem remover
  - tamanho(P): retorna o número de elementos em P
  - vazia(P): indica se a pilha P está vazia
  - cheia(P): indica se a pilha P está cheia (útil para implementações sequenciais).

Implementação Sequencial

- Implementação simples
- Uma variável mantém o controle da posição do topo, e pode ser utilizada também para informar o número de elementos na pilha (tamanho)

Definição da Interface das Operações

- PILHA *pilha_criar(void);
- void pilha_apagar(PILHA **pilha);
- int pilha_vazia(PILHA *pilha);
- int pilha_cheia(PILHA *pilha);
- int pilha_tamanho(PILHA *pilha);
- ITEM *pilha_topo(PILHA *pilha);
- int pilha_empilhar(PILHA *pilha, ITEM *item);
- ITEM *pilha_desempilhar(PILHA *pilha);
- void pilha_print(PILHA *p);
- void pilha_inverter(PILHA *p);

Definição de Tipos

```
/* interface (arquivo .h). Ex.: pilha.h*/
1  #include “item.h”
2  typedef struct pilha PILHA;
3
4  #define TAM 100
5  #define ERRO -32000
...

/* implementação (arquivo .c). Ex.: pilha.c*/
1  struct pilha {
2     ITEM *item[TAM];
3     int topo;
4  };
5 ...
```

Definição de Tipos

```
/* programa cliente (arquivo .c). Ex.: decimal_binario.c*/
1  #include “pilha.h”
2 ...
30 PILHA *p;
31...
```

Implementação Sequencial

```
PILHA* pilha_criar(void) {
    PILHA* pilha = (PILHA *) malloc(sizeof (PILHA));
    if (pilha != NULL)
        pilha->topo = 0;
    return (pilha);
}
int pilha_vazia(PILHA* pilha) {
    return ((pilha != NULL) ? pilha->topo == 0 : ERRO);
}
int pilha_cheia(PILHA *pilha) {
    return ((pilha != NULL) ? pilha->topo == TAM : ERRO);
}
int pilha_tamanho(PILHA *pilha) {
    return ((pilha != NULL) ? pilha->topo : ERRO);
}
```

Implementação Sequencial

```
int pilha_empilhar(PILHA *pilha, ITEM* item) {
     if ((pilha!=NULL) && (!pilha_cheia(pilha)) {
          pilha->item[pilha->topo] = item;
          pilha->topo++;
          return (1);
     }
     return (ERRO);
}
ITEM* pilha_desempilhar(PILHA* pilha) {
     ITEM* i;
     if ((pilha != NULL) && (!pilha_vazia(pilha))) {
          i = pilha_topo(pilha);
          pilha->item[pilha->topo] = NULL;
          pilha->topo--;
          return (i);
     }
     return (NULL);
}
```

Exemplo prático (para quem quiser)

- Implemente um programa em C para converter um número decimal em sua respectiva representação binária usando o TAD pilha (sequencial).

Referências

- ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.