Programação I

Tipos Abstratos de Dados
Stacks e Queues

Estrela Ferreira Cruz

1

Objetivos da aula

Objetivos da aula:

Tipos abstratos de dados:

Stacks ou Pilhas

- Apresentação do conceito e modo de funcionamento;
- Operações sobre Stacks;
- Apresentação de exemplos práticos;
- Implementação de uma stack recorrendo ao uso de um array;

Queues ou filas:

- Apresentação do conceito e princípio de funcionamento.
- Operações com Queues.
- Apresentação de exemplos práticos;
- Implementação recorrendo ao uso de listas duplamente ligadas.

2

Tipos abstratos de dados

Tipos de dados

A representação real dos dados num computador:

- Num nível mais baixo, começamos com os tipos de dados básicos como o char, int, float ou double.
- No nível seguinte estão as estruturas homogéneas (arrays), que são conjuntos organizados de dados todos do mesmo tipo.
- Depois estão as estruturas (registos), que agrupam tipos de dados diferentes que podem ser acedidos por um único nome.
- Os tipos abstratos de dados (TAD), além dos dados a armazenar trata também das operações sobre os dados.

Estrela F. Cruz

3

Tipos abstratos de dados

Tipos abstratos de dados

- Um Tipo Abstratos de Dados (TAD) é um tipo de dados que é organizado de modo a que a especificação dos dados e as operações sobre esses dados seja separada da representação e da implementação das operações.
- Transcendendo os aspetos físicos dos dados, o nível final concentra-se na sequência pela qual os dados são acrescentados e acedidos, coexistindo na sua definição uma estrutura lógica.

Estrela F. Cruz

4

Tipos abstratos de dados

Vantagens de uso de tipos abstratos de dados

- Facilita o encapsulamento e segurança: o utilizador não tem acesso direto aos dados. O acesso é feito através de um conjunto de operações de acesso aos dados, previamente definidas.
- A estratégia de implementação pode ser "escondida".
- A implementação é separada da utilização.
- Facilita a manutenção e a reutilização do código.

Como exemplo de tipos de dados abstratos temos:

- Stack ou pilhas
- Queues ou filas

Estrela F. Cruz

5

Stacks

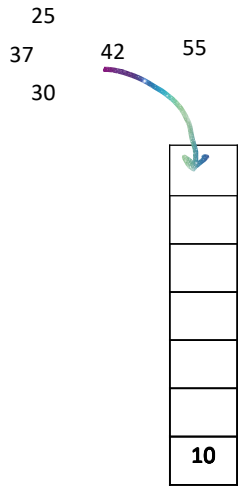
Stacks ou Pilhas

- É uma estrutura de dados em que a inserção e a remoção de elementos se faz pela mesma extremidade (o ultimo a entrar é o primeiro a sair) geralmente designada por topo da stack.
- Princípio de inserção/remoção é LIFO (Last-In First-Out),
- Os elementos podem ser inseridos sem restrições mas só o inserido em último lugar é que pode ser removido a qualquer momento.
- Podemos ver o funcionamento de uma pilha na pagina seguinte

Estrela F. Cruz

6

Stacks

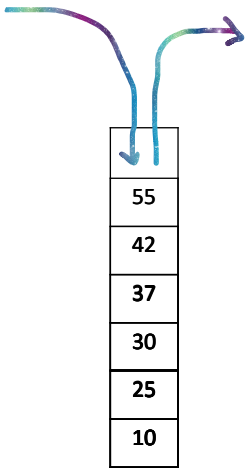


Stack

Estrela F. Cruz

7

Stacks

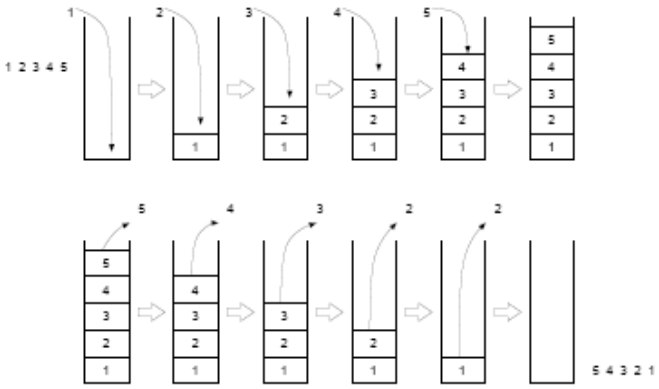


Estrela F. Cruz

8

Stacks

Operações com pilhas



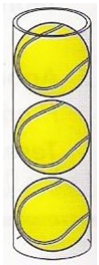
Estrela F. Cruz

9

Stacks

Exemplos dia a dia:

- Um exemplo de uma stack é uma pilha de papéis na secretária. Os papéis são colocados uns por cima dos outros e o primeiro a ser retirado é o de cima, ou seja, o último a ser colocado.
- Outro exemplo é uma pilha de pratos. O prato que serve de base é o último a ser usado e o prato do topo, que é o último a ser colocado na stack, é o primeiro a ser usado.
- Uma stack (pilha) de cadeiras.
- Bolas de ténis armazenadas num recipiente próprio: a primeira a entrar para o recipiente é sempre a última a sair.
- Etc.



Estrela F. Cruz

10

Stacks

Exemplos de stacks no contexto Software:

- A stack é adequada para guardar contextos, mantendo disponível o contexto corrente, ou seja, o mais atual.
- As stacks são normalmente usadas em software de sistemas, incluindo compiladores e interpretadores. A maioria dos compiladores de C usa uma stack na passagem de argumentos para as funções.
- As stacks são também usadas pelos compiladores para verificar a correta utilização dos (,{ , " e respetivos),},", comentários (/*… */), etc.
- As stacks são usadas por editores (por ex. MS Word) para permitirem o "undo" das operações. A última operação a ser executada é a primeira a ser desfeita.
- As stacks são também usadas pelos browsers para permitirem o recuo (backtracking) nas páginas visitadas.
- Etc.

Estrela F. Cruz

11

Stacks

Operações sobre a stack:

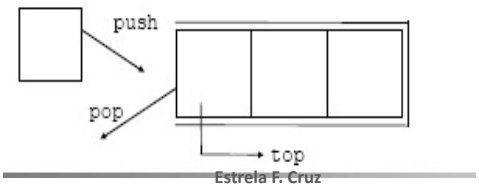
push - Insere um elemento no topo da stack

pop - Retira o elemento do topo da stack

top - Lê e retorna o valor do elemento do topo da stack, mas não o retira da stack.

isEmpty - Retorna Verdadeiro se a stack está vazia, noutro caso retorna falso.

size - Retorna o numero de elementos da stack.



Estrela F. Cruz

12

Stacks

Exercício: Indique o resultado após cada uma das seguintes operações e os respetivos efeitos numa pilha de números inteiros inicialmente vazia:

Instrução	Descrição	Stack (composição)
1.push(8)	insere o valor 8	8
2.push(3)	insere o valor 3	83
3.pop()	retira o valor 3	8
4.push(7)	insere o valor 7	87
5.pop()	retira o valor 7	8
6.top()	retorna o valor 8	8
7.pop()	retira valor 8	vazio
8.pop()	retira "Satck vazia"	vazio
9.isEmpty()	retorna Verdadeiro	vazio
10.push(9)	insere o valor 9	9
11.push(5)	insere o valor 5	95
12.size()	retorna 2	95
13.isEmpty()	retorna Falso	95

Estrela F. Cruz

13

Stacks

Implementação da Stack

- Podemos optar por uma gestão mais rígida - baseada em arrays (vetores) - tem como desvantagem o facto de ser necessário conhecer o tamanho máximo à partida. O espaço ocupado em memória corresponde sempre ao tamanho máximo.
- Ou por uma gestão mais flexível - com listas ligadas- tem como desvantagem o facto de ser menos eficiente (maior tempo de computação) mas permite o processamento de problemas maiores em resultado da gestão criteriosa da memória.
- Do ponto de vista do utilizador, é irrelevante a forma como a stack é implementada, no entanto, se for implementada com alocação dinâmica de memória, a stack deve aceitar sempre elementos, a menos que existam problemas na alocação de memória.

Estrela F. Cruz

14

Stacks

Implementação da Stack em vetores:

- É necessário uma variável, à qual foi atribuído o nome tos (TOpo daStack), que é o índice da próxima posição livre da stack.
- É necessário verificar que não se ultrapassa o limite da stack no armazenamento (stack overflow) e que não se tente retirar dados com a stack vazia (stack underflow).
- Ou seja: se tos é igual a 0, a stack está vazia e se tos é maior que a última posição de memória, ou seja, o tamanho máximo do vetor, a stack está cheia (stack overflow) .
- Em seguida podemos ver um exemplo de implementação de uma stack de caracteres recorrendo ao uso de um array estático.

Estrela F. Cruz

15

Stacks

```
int size(int tos){
    return tos;
}

int isEmpty(int tos){
    return (tos==0);
}

char pop(char stack[], int *tos){
    char aux='\0';
    if((*tos)<=0){
      printf("Stack underflow\n");
    }
    else {
        aux=stack[(*tos)-1];
        (*tos)--;
    }
    return aux;
}
```

```
void push(char stack[], char car,
int *tos){
    if((*tos) >= MAX){
       printf("Stack Overflow\n");
       return;
    }
    stack[*tos]=car;
    (*tos)++;
}

char top(char stack[], int tos){
    char aux='\0';
    if(tos <=0){
        printf("stack vazia\n");
    }
    else {
        aux=stack[tos-1];
    }
    return aux;
}
```

Estrela Ferreira Cruz

Estrela F. Cruz

16

Stacks

Implementação da Stack em Listas Ligadas:

Uma stack pode ser implementada usando uma lista ligada onde a inserção e a remoção se fazem na mesma extremidade da lista:

- Inserção e remoção no inicio da lista, ou seja, o último a inserir é o primeiro a ser removido – LIFO.
- Inserção e remoção no fim da lista, da mesma forma o último a ser inserido é o primeiro a ser removido – LIFO.

Estrela F. Cruz

17

Queues ou filas

Queues ou Filas:

- Princípio de inserção/remoção: FIFO (First-in First-out), ou seja, o primeiro a entrar é o primeiro a sair.
- A inserção e a remoção de elementos de uma queue faz-se por extremidades opostas, geralmente designadas por cabeça e cauda.
- Numa queue, a única forma de inserir um elemento é no fim da fila.
- Apenas se consegue obter, ou ler, o elemento que está no início da fila.
- Não é permitido o acesso de algum elemento em particular, no meio da queue.

Estrela F. Cruz

18

Queues ou filas

Existem muito exemplos práticos de filas no dia-a-dia:

- Uma fila para o bar, onde os primeiros a chegar são o primeiros a ser atendidos.
- Uma fila de pessoas para entrar num teatro. As pessoas, à medida que vão chegando vão para o fundo da fila. As pessoas que se vão sentando, saem da frente da fila.
- O buffer de uma impressora para pedidos com a mesma prioridade, os primeiros pedidos a chegar são os primeiros a ser impressos.
- Armazenamento de I/O (caracteres escritos no teclado).

Estrela F. Cruz

19

Queues ou filas

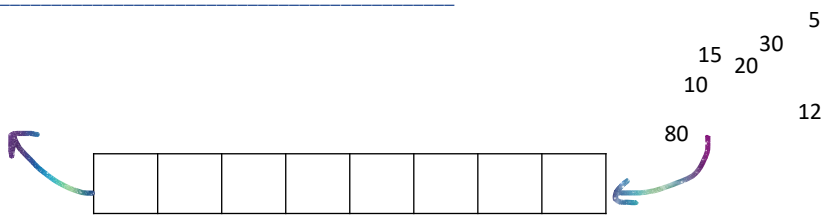
Exemplos de aplicações que recorrem ao uso de queues:

- Programas de distribuição de eventos que permite adicionar um certo número de eventos, podendo servir, por exemplo, para organizar compromissos diários. Cada evento executado é retirado da fila e o próximo evento é visualizado.
- Programas de gestão dos aviões numa pista para levantar voo.
- Programas de gestão de reservas.
- Programas de gestão de chamadas telefónicas.
- Etc.

Estrela F. Cruz

20

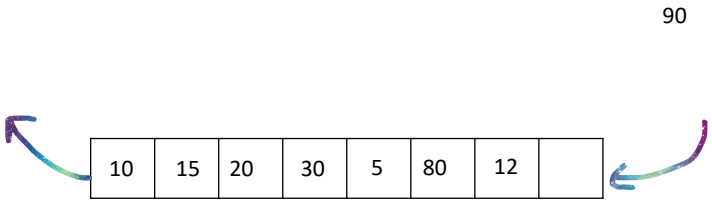
Queues ou filas



Estrela F. Cruz

21

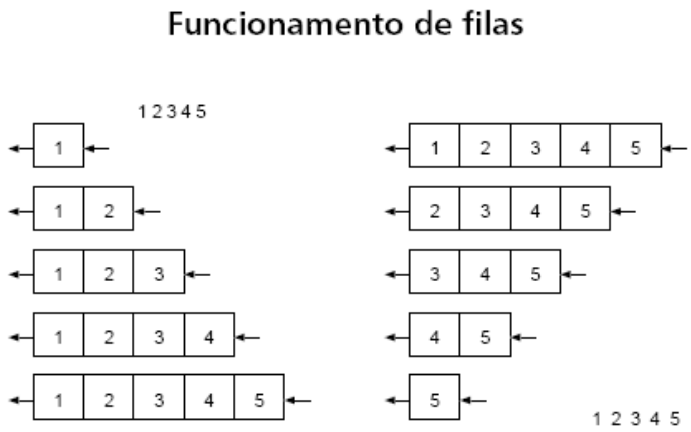
Queues ou filas



Estrela F. Cruz

22

Queues ou filas



Estrela F. Cruz

23

Queues ou filas

Operações sobre queues:

enqueue - Insere um elemento na fim da fila

dequeue - Retira o elemento da frente da fila

front - Lê e retorna o valor do elemento da frente da fila, mas não o retira.

IsEmpty - Retorna Verdadeiro de a fila está vazia, noutro caso retorna Falso.

size - Retorna o numero de elementos da fila.

NOTA: a operação enqueue pode também ser chamada de qstore e a operação dequeue pode ser tratada por qretrieve.

Estrela F. Cruz

24

Queues ou filas

Exercício - Indique o resultado após cada uma das seguintes operações e os respetivos efeitos numa queue de números inteiros inicialmente vazia:

Instrução	Descrição	Qqueue (conteúdo)
1. enqueue(3)	insere o valor 3	3
2. enqueue(7)	insere o valor 7	37
3. dequeue()	retira o valor 3	7
4. enqueue(9)	insere o valor 9	79
5. dequeue()	retira o valor 7	9
6. front()	retorna o valor 9	9
7. dequeue()	retira valor 9	vazio
8. isEmpty()	retorna Verdadeiro	vazio
9. enqueue(2)	insere o valor 2	2
10. enqueue(9)	insere o valor 9	29
11. enqueue(5)	insere o valor 5	295
12. size()	retorna 3	295
13. isEmpty()	retorna Falso	295

Estrela F. Cruz

25

Queues ou filas

Implementação da Queue:

Tal como nas stacks, também nas queues podemos optar por uma gestão mais rígida, baseada em vetores, ou por uma gestão mais flexível, usando alocação dinâmica de memória.

- As vantagens e desvantagens de cada uma das opções são idênticas às da stack.
- Do ponto de vista do utilizador, é irrelevante a forma como a queue é implementada.

No caso de ser implementada com vetores, é necessário ter em atenção o limite máximo do vetor.

Estrela F. Cruz

26

Queues ou filas

Implementação da queue num vetor estático.

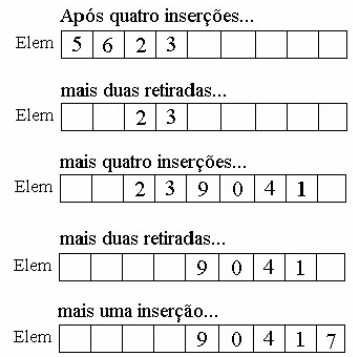
- Temos que guardar não só a posição da frente da fila, de onde vamos retirar elementos, como também o inicio da fila, onde os elementos vão ser inseridos.
- Tendo as variáveis topPos e backPos, onde topPos contem o índice da próxima posição livre de armazenamento e backPos contem o índice do próximo elemento a retirar ou ler.
- De salientar que se backPos for igual a topPos não existem mais elementos na fila.
- A função enqueue (qstore) deve verificar se a fila atingiu, ou não, o limite máximo antes de colocar um novo elemento na fila.
- A função dequeue (qretrieve) deve verificar se a fila está, ou não, vazia antes de retirar um elemento da lista.
- Quando um novo elemento é colocado na fila, topPos é incrementado.
- Quando um elemento é retirado, backPos é incrementado.

Estrela F. Cruz

27

Queues ou filas

Implementando-se as filas com continuidade, ocorrerá a seguinte situação, após inserir e retirar vários elementos:



Estrela F. Cruz

28

Queues ou filas

Implementação da Queues em Listas Ligadas

- Uma Queue, tal como a Stack, pode ser implementada usando uma lista simples ou duplamente ligada. No caso das queues a inserção e a remoção fazem-se em extremidades opostas:
  - Inserção no inicio e remoção no fim, ou seja, o primeiro a inserir é o primeiro a sair – FIFO.
  - Inserção no fim e remoção no inicio da lista, da mesma forma o primeiro a inserir é o primeiro a sair – FIFO.
- A implementação de Queues em listas duplamente ligadas tem a vantagem de serem conhecidos o inicio e o fim da lista, sendo por isso mais fácil de gerir.

Estrela F. Cruz

29

Queues ou filas

Queues circulares

- Quando a implementação da queue é feita recorrendo ao uso de um array estático, numa situação final, o array ainda tem espaço mas se o espaço libertado não é reocupado, o fim do array pode ser atingido com alguma facilidade;
- Como a fila tem o comportamento típico de se "deslocar" para o final do array, podemos tentar reutilizar o espaço que está efetivamente disponível.
- A solução mais comum (quando se está a implementar a fila recorrendo ao uso de um array estático) é recorrer ao uso de arrays circulares ou filas circulares.

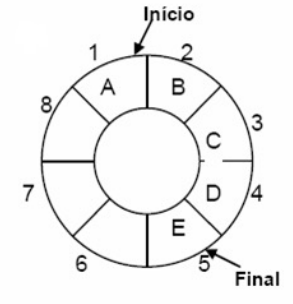
Estrela F. Cruz

30

Queues ou filas

Queues circulares (ou arrays circulares)

- De modo a otimizar o armazenamento de dados e respetiva gestão de espaço na memória do computador, as filas circularem seguem o mesmo principio das filas simples, mas têm uma gestão memória mais eficaz.
- Nas filas simples apenas se podem inserir elementos novamente no inicio da fila, quando esta fica vazia.
- As filas circulares podem ser representadas da forma como se vê na figura do lado.



Estrela F. Cruz

31

Bibliografia

- Programação Avançada Usando C, António Manuel Adrego da Rocha, ISBN: 978-978-722-546-0.
- Schildt, Herbert: C the complete Reference, McGraw-Hill, 1998.
- Algoritmia e Estruturas de Dados, José Braga de Vasconcelos, João Vidal de Carvalho, ISBN: 989-615-012-5.
- Elementos de Programação com C - Pedro João Valente D. Guerreiro, 3ª edição, ISBN: 972-722-510-1.
- Introdução à Programação Usando C, António Manuel Adrego da Rocha, ISBN: 972-722-524-1.

Estrela F. Cruz

32