Alocação Encadeada

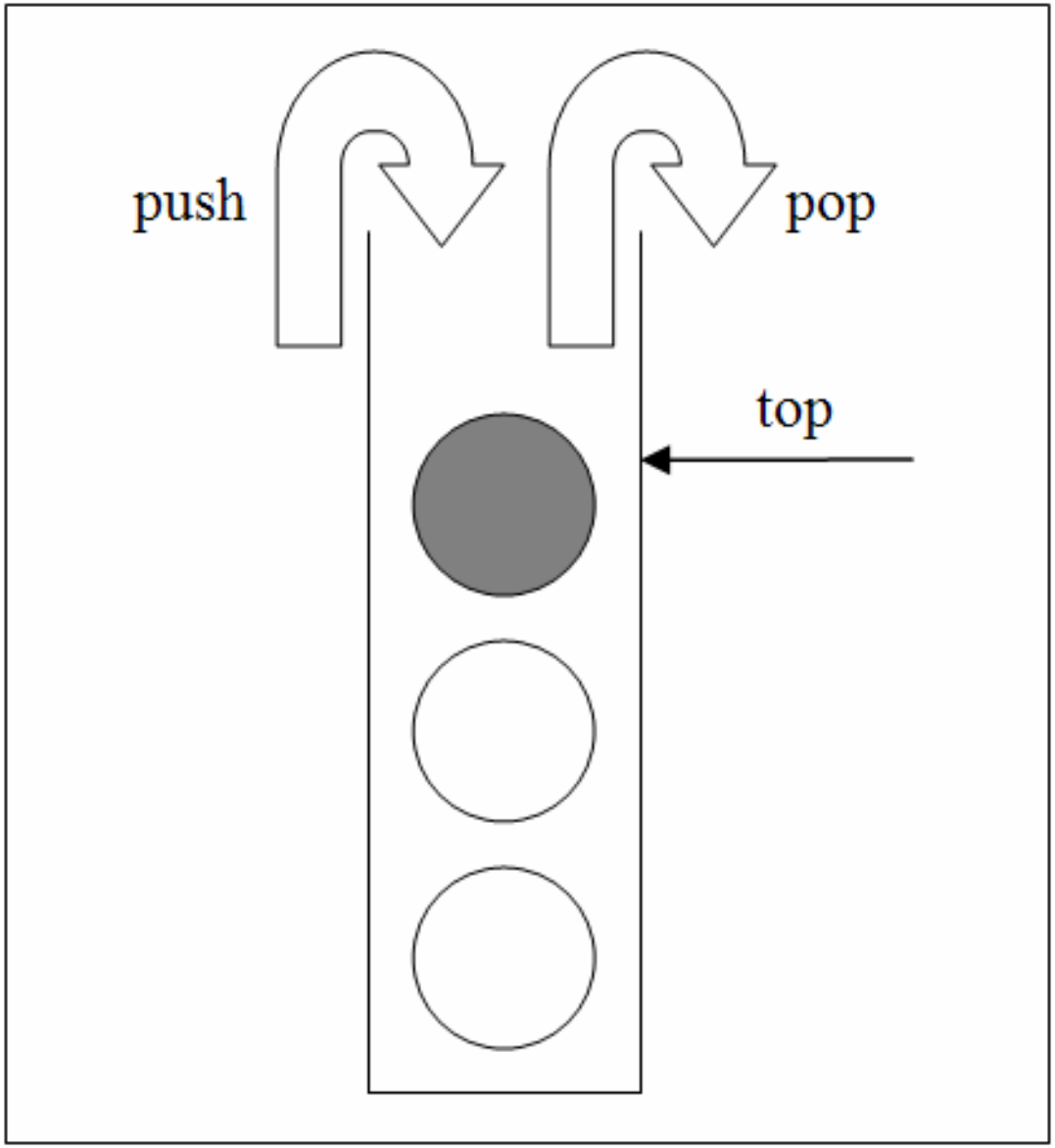
Como vimos, uma fila nada mais é do que uma lista com uma disciplina de acesso. Logo, podemos nos utilizar de todos os conceitos vistos em listas para implementarmos filas. Por exemplo, podemos utilizar uma lista circular para armazenar uma fila.

Pilhas

Caracterização

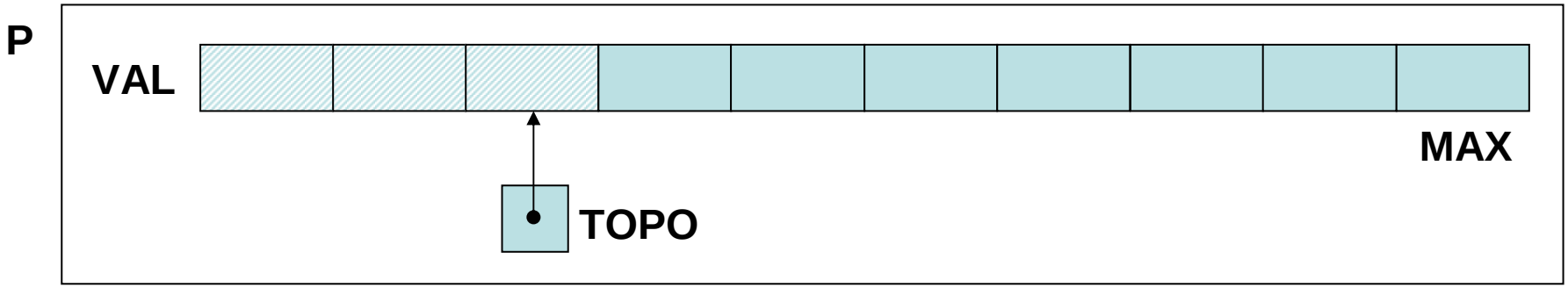
Uma pilha é uma lista com restrições de acesso, em que todas as operações só podem ser aplicadas sobre uma das extremidades, o topo da pilha. Com isso estabelece-se o critério LIFO (last in, first out), que indica que o último item que entra é o primeiro a sair. O modelo intuitivo para isto é o de uma pilha de pratos, ou livros, etc, na qual só se pode visualizar (consultar) o último empilhado e este é o único que pode ser retirado. E também qualquer novo empilhamento (inserção) se fará sobre o último da pilha.

Caracterização



Alocação Seqüencial

Uma forma de se implementar uma pilha é armazená-la num vetor VAL de MAX valores associado com um cursor inteiro TOPO que indica onde está o topo da pilha, como se vê abaixo:



A Implementação das operações é trivial: a pilha cresce do começo para o fim do vetor; uma pilha vazia tem o

Alocação Seqüencial

cursor TOPO igual a -1; a cada inserção, o cursor topo é incrementado para apontar para a próxima posição livre do vetor, onde é armazenado o novo elemento; uma retirada decrementa o cursor; uma consulta devolve o valor do elemento indexado por TOPO.

De pose destas informações definiremos e implementaremos agora o TAD PILHA_SEQ:

```
typedef struct
{
   int TOPO; /*índice do topo da pilha*/
   int VAL[MAX];    /*vetor de elementos*/
}PILHA_SEQ;
void cria_pilha (PILHA_SEQ *);
int eh_vazia (PILHA_SEQ *);
void push (PILHA_SEQ *, int);
int top (PILHA_SEQ *);
void pop (PILHA_SEQ *);
int top_pop (PILHA_SEQ *);
```

```
void cria_pilha (PILHA_SEQ *p)
{
   p->TOPO = -1;
}

int eh_vazia (PILHA_SEQ *p)
{
   return (p->TOPO == -1);
}
```

```
void push (PILHA_SEQ *p, int v)
{
   if (p->TOPO == MAX-1)
   {
      printf ("\nERRO! Estouro na pilha.\n");
      exit (1);
   }
   p->VAL[++(p->TOPO)]=v;
}
```

```
int top (PILHA_SEQ *p)
{
   if (eh_vazia(p))
   {
      printf ("\nERRO! Consulta na pilha vazia.\n");
      exit (2);
   }
   else
      return (p->VAL[p->TOPO]);
}
```

```
void pop (PILHA_SEQ *p)
{
   if (eh_vazia(p))
   {
      printf ("\nERRO! Retirada na pilha vazia.\n");
      exit (3);
   }
   else
      p->TOPO--;
}
```

```
int top_pop (PILHA_SEQ *p)
{
   if (eh_vazia(p))
   {
      printf ("\nERRO! Consulta e retirada na pilha
   vazia.\n");
      exit (4);
   }
   else
      return (p->VAL[p->TOPO--]);
}
```

Alocação Seqüencial - Exercício

Utilizando-se dos TAD´s FILA_SEQ e PILHA_SEQ, vistos anteriormente, implemente a seguinte operação:

void inverte_fila (FILA_SEQ *f);

a qual recebe uma referência para uma fila seqüencial de inteiros e inverte a ordem de seus elementos, utilizando-se para isto de uma pilha seqüencial.