Filas – Exemplo de Aplicação

Utilizando-se dos TAD´s LISTA_ENC e FILA_ENC, implemente a função ord_por_dist() a qual recebe como entrada uma referência para uma lista encadeada de valores inteiros e a ordena através do processo de ordenação por distribuição.

OBS: b=10 e d=4.

```c
void ord_por_dist (LISTA_ENC *pl) {
    int c1, c2;
    FILA_ENC vf[b];
    for (c2=0; c2<b ; c2++)
        cria_fila(&vf[c2]);
    for (c1=0; c1<d; c1++) {
        while (!eh_vazia_l(*pl)) {
            int v=recup(*pl,1),i;
            i=(v/((int)(pow (b, c1))))%b;
            ins(vf[i],v);
            ret_l(pl,1);
        }
        for (c2=0; c2<b ; c2++)
            while (!eh_vazia(vf[c2]))
                ins_l(pl,cons_ret(vf[c2]) , tam_l(*pl)+1);
    }
}
```

Pilhas
Alocação Sequencial

Pilha - Caracterização

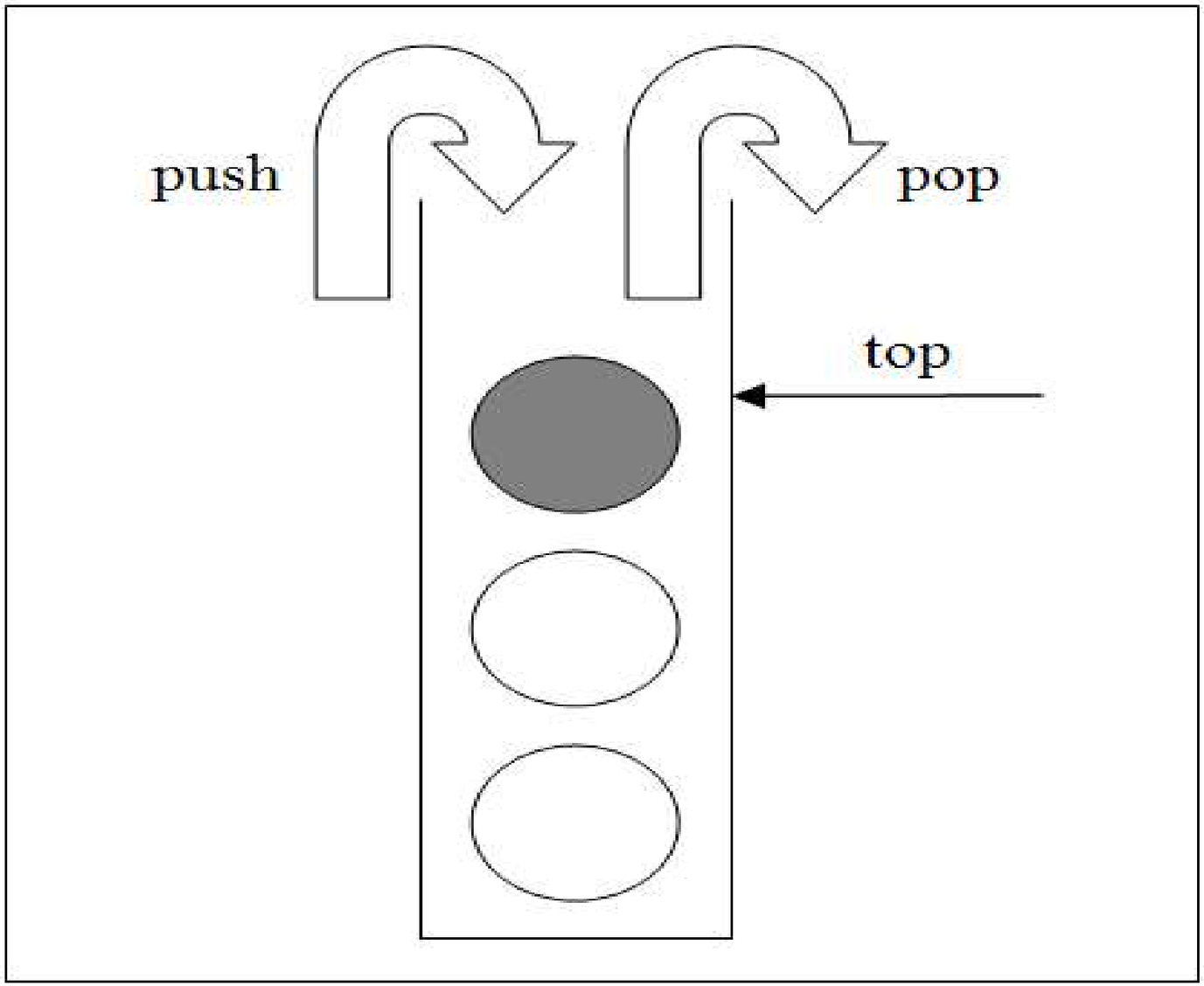
Uma pilha é uma lista com restrições de acesso, onde todas as operações só podem ser aplicadas sobre uma das extremidades da lista, denominada topo da pilha.

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, por exemplo, uma pilha de livros onde só se pode visualizar o último empilhado e este é o único que pode ser retirado; Qualquer novo empilhamento se fará sobre o último da pilha.

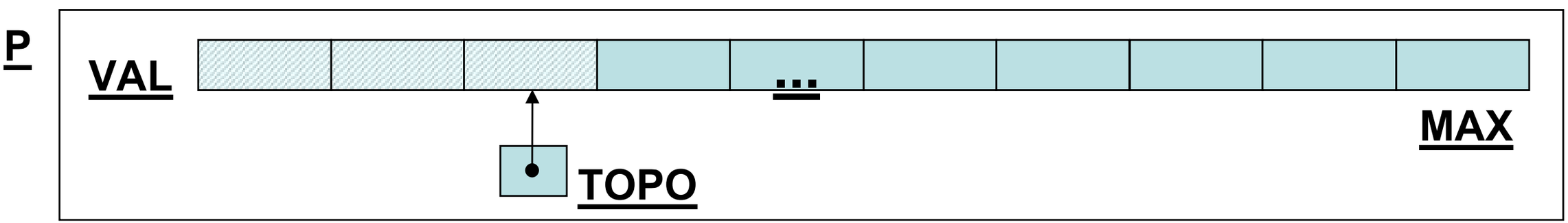
UNIVASF

Pilha - Caracterização



Pilha – Alocação Sequencial

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



De posse destas informações definiremos e implementaremos agora o TAD PILHA_SEQ de valores inteiros.

```c
typedef struct
{
    int TOPO;
    int VAL[MAX];
}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 *);
```

Pilha - Alocação Sequencial

Com base no que foi visto implemente a operação cria_pilha() que compõem o TAD PILHA_SEQ.

```
1   typedef struct
2   {
3       int TOPO;
4       int VAL[MAX];
5   }PILHA_SEQ;
6   void cria_pilha (PILHA_SEQ *);
7   int eh_vazia (PILHA_SEQ *);
8   void push (PILHA_SEQ *, int);
9   int top (PILHA_SEQ *);
10  void pop (PILHA_SEQ *);
11  int top_pop (PILHA_SEQ *);
```

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

Pilha - Alocação Sequencial

Com base no que foi visto implemente a operação eh_vazia() que compõem o TAD PILHA_SEQ.

```
1   typedef struct
2   {
3       int TOPO;
4       int VAL[MAX];
5   }PILHA_SEQ;
6   void cria_pilha (PILHA_SEQ *);
7   int eh_vazia (PILHA_SEQ *);
8   void push (PILHA_SEQ *, int);
9   int top (PILHA_SEQ *);
10  void pop (PILHA_SEQ *);
11  int top_pop (PILHA_SEQ *);
```

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

Pilha - Alocação Sequencial

Com base no que foi visto implemente a operação push() que compõem o TAD PILHA_SEQ.

```
1  typedef struct
2  {
3      int TOPO;
4      int VAL[MAX];
5  }PILHA_SEQ;
6  void cria_pilha (PILHA_SEQ *);
7  int eh_vazia (PILHA_SEQ *);
8  void push (PILHA_SEQ *, int);
9  int top (PILHA_SEQ *);
10 void pop (PILHA_SEQ *);
11 int top_pop (PILHA_SEQ *);
```

```c
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;
}
```

Pilha - Alocação Sequencial

Com base no que foi visto implemente a operação top() que compõem o TAD PILHA_SEQ.

```
1   typedef struct
2   {
3       int TOPO;
4       int VAL[MAX];
5   }PILHA_SEQ;
6   void cria_pilha (PILHA_SEQ *);
7   int eh_vazia (PILHA_SEQ *);
8   void push (PILHA_SEQ *, int);
9   int top (PILHA_SEQ *);
10  void pop (PILHA_SEQ *);
11  int top_pop (PILHA_SEQ *);
```

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

Pilha - Alocação Sequencial

Com base no que foi visto implemente a operação pop() que compõem o TAD PILHA_SEQ.

```
1   typedef struct
2   {
3       int TOPO;
4       int VAL[MAX];
5   }PILHA_SEQ;
6   void cria_pilha (PILHA_SEQ *);
7   int eh_vazia (PILHA_SEQ *);
8   void push (PILHA_SEQ *, int);
9   int top (PILHA_SEQ *);
10  void pop (PILHA_SEQ *);
11  int top_pop (PILHA_SEQ *);
```

```
1   void pop (PILHA_SEQ *p)
2   {
3       if (eh_vazia(p))
4       {
5           printf ("\nERRO! Retirada na pilha vazia.\n");
6           exit (3);
7       }
8       else
9           p->TOPO--;
10  }
```

Pilha - Alocação Sequencial

Com base no que foi visto implemente a operação top_pop() que compõem o TAD PILHA_SEQ.

```
1   typedef struct
2   {
3       int TOPO;
4       int VAL[MAX];
5   }PILHA_SEQ;
6   void cria_pilha (PILHA_SEQ *);
7   int eh_vazia (PILHA_SEQ *);
8   void push (PILHA_SEQ *, int);
9   int top (PILHA_SEQ *);
10  void pop (PILHA_SEQ *);
11  int top_pop (PILHA_SEQ *);
```

```c
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--]);
}
```