

Problema

➤ Conhecem a brincadeira Pato-Pato-Ganso?

Sentar em círculo



Escolher quem será o “pegador” ou o ganso



Ande ao redor do círculo, tocando as cabeças para escolher um ganso



Pegador começa a correr ao redor do círculo, enquanto o ganso se levanta e corre atrás dele. O objetivo do ganso é alcançar o pegador antes que ele se sente no lugar do ganso. Se o ganso alcança o pegador, o ganso se torna o novo pegador e o pegador sai do círculo.



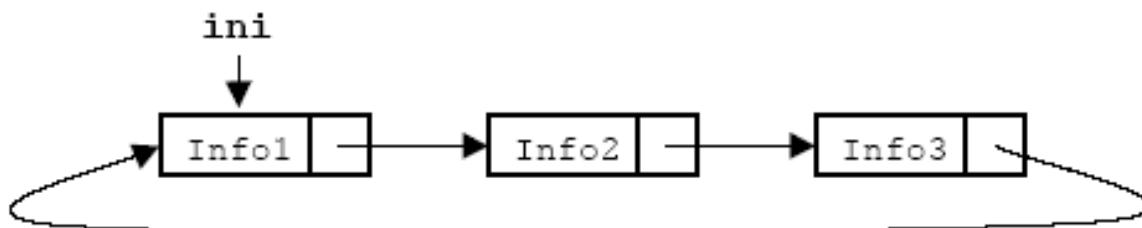
Problema

- Como podemos fazer um programa para controlar essa brincadeira?
- Como as crianças deveriam ser armazenadas no nosso programa?
- Como deveria ser a inclusão e exclusão de crianças da brincadeira?

Listas Circulares

Definição

- Numa lista circular, o último elemento tem como próximo o primeiro elemento da lista, formando um ciclo.
- A rigor, neste caso, não faz sentido falarmos em primeiro ou último elemento.
- A lista pode ser representada por um ponteiro para um elemento inicial qualquer da lista.

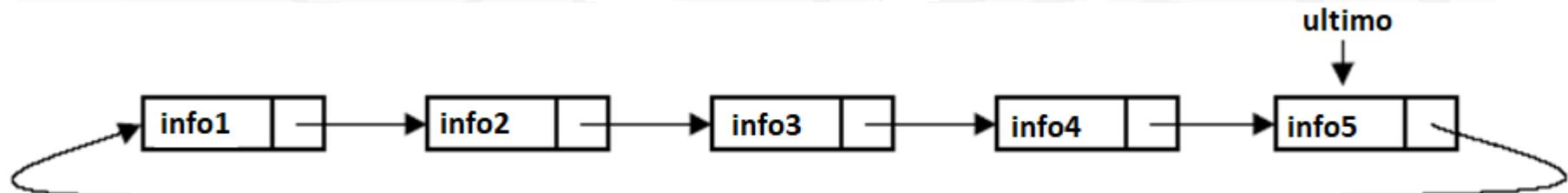


Impressão

- Pense em como poderia ser a impressão em uma lista circular.

Convenção

- Uma convenção interessante numa lista circular é guardarmos o endereço do último nó ao invés do primeiro, uma vez que para saber qual é o primeiro basta pegar o endereço do nó seguinte ao último :



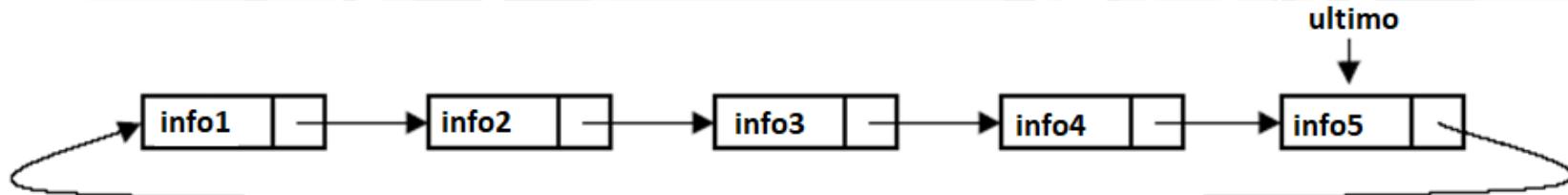
Convenção

- Esta convenção tem a vantagem de poder incluir ou remover um elemento convenientemente a partir do início ou final de uma lista.
- Além disso, é estabelecida a convenção de que um ponteiro nulo representa uma lista circular vazia.



Função para a Impressão

```
void imprime_circular (PLista ultimo) {  
    PLista p;  
    if (ultimo !=NULL) {  
        p = ultimo->prox;  
        /* percorre os elementos até alcançar novamente o  
        início */  
        do {  
            printf("%d\n", p->info);  
            p = p->prox;  
        } while (p != ultimo->prox);  
    }  
}
```



Inserção em Lista Circular

- Faça a inserção em lista circular na primeira posição da lista. Lembre-se que a inserção em lista pode ser feita inserindo em qualquer lugar.

```

typedef struct lista {
    int info;
    struct lista* prox;
}TLista;
typedef TLista *PLista;

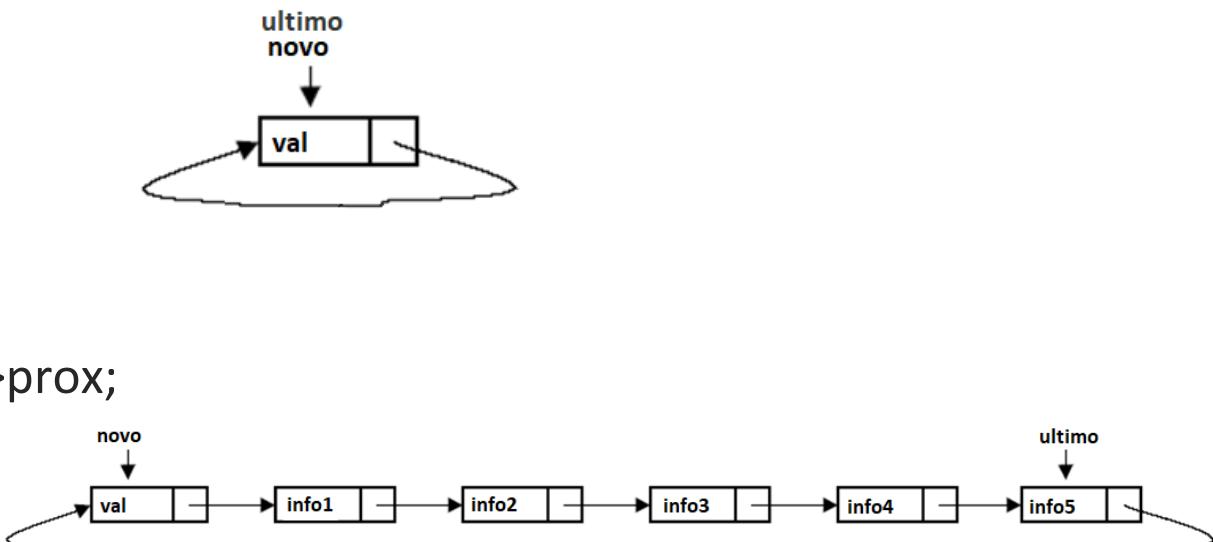
```

/ inserção no início: retorna a lista atualizada */*

```

PLista insere (PLista ultimo, int val){
    PLista novo = (PLista) malloc(sizeof(TLista));
    novo->info = val;
    if (ultimo == NULL){
        ultimo = novo;
        novo->prox = novo;
    }
    else{
        novo->prox = ultimo->prox;
        ultimo->prox = novo;
    }
    return ultimo;
}

```



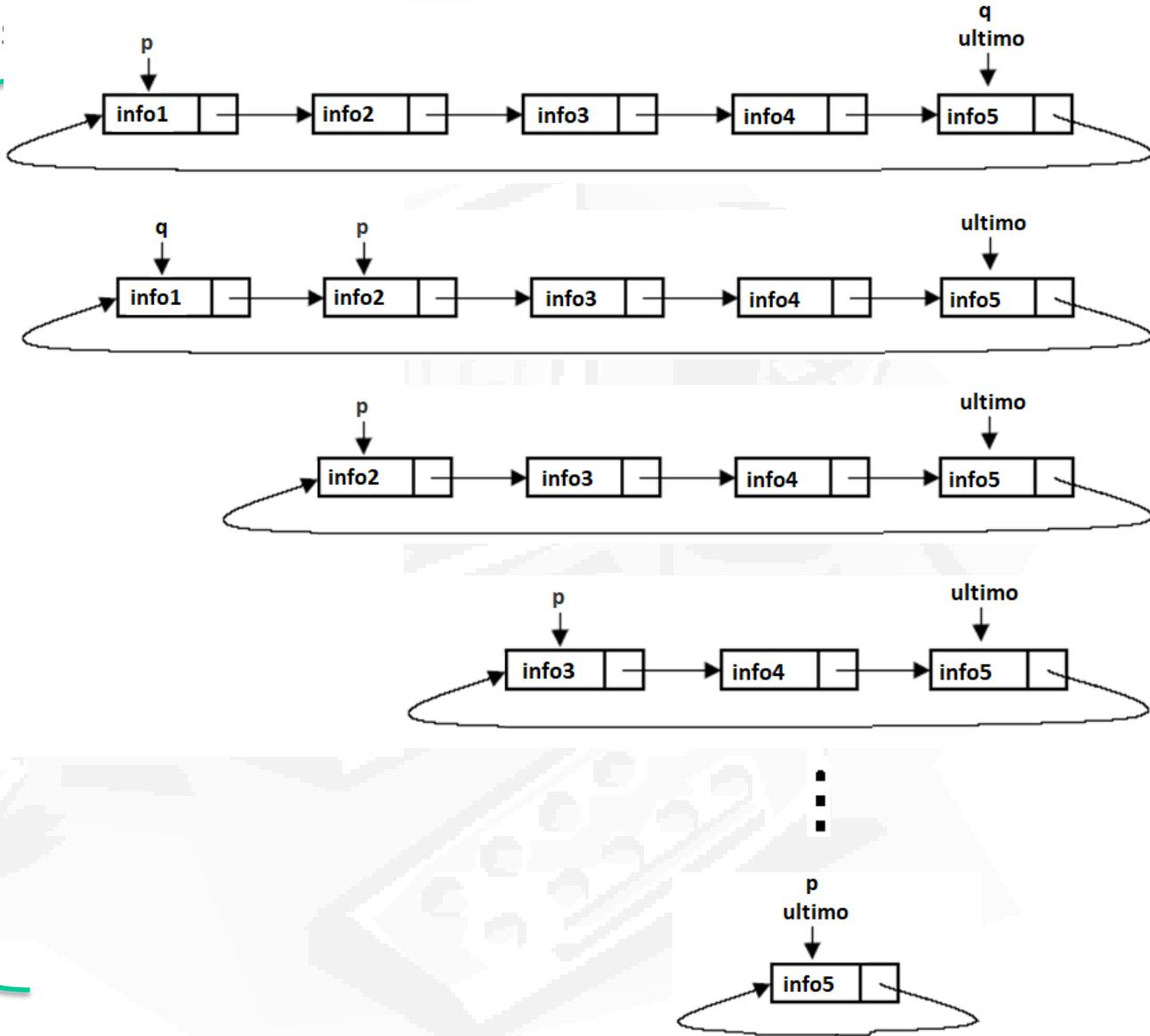
Liberação em Lista Circular

- Faça uma função para liberar todo o espaço alocado em uma lista circular.

```

/* libera os espaços alocados
void libera (PLista ultimo){
    PLista p = ultimo->prox;
    PLista q = ultimo;
    while (p != ultimo){
        q = p;
        p = p->prox;
        free(q);
    }
    free(ultimo);
}

```



Exercício 1

- Faça as funções Push () e Pop() utilizando listas circulares. Considere *ultimo* um ponteiro para o último nó da lista circular e considere que o primeiro nó seja o topo da pilha.

```
/* Push: igual à inserção em lista circula já vista,  
ou seja, insere no início da lista, guardando o endereço do último da lista */
```

```
PLista Push (PLista ultimo, int val){
```

```
    PLista novo = (PLista) malloc(sizeof(TLista));
```

```
    novo->info = val;
```

```
    if (ultimo == NULL){
```

```
        ultimo = novo;
```

```
        novo->prox = novo;
```

```
}
```

```
    else{
```

```
        novo->prox = ultimo->prox;
```

```
        ultimo->prox = novo;
```

```
}
```

```
    return ultimo;
```

```
}
```

```
/* Pop: como é pilha, se insere no início, retira do início também */
```

```
PLista Pop (PLista ultimo, int *val){
```

```
PLista p;
```

```
if (ultimo == NULL)
```

```
    printf("Pilha vazia!!\n\n");
```

```
else{
```

```
    p = ultimo->prox; // p recebe o 1o.elemento
```

```
    *val = p->info;
```

```
    ultimo->prox = p->prox;
```

```
    if (p==ultimo)
```

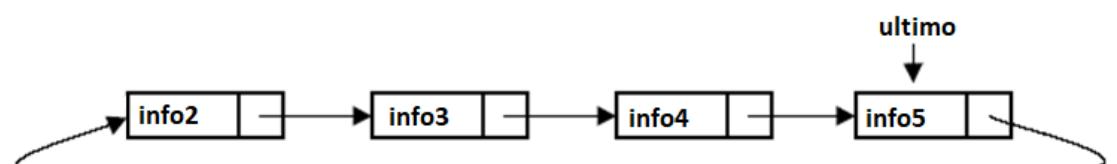
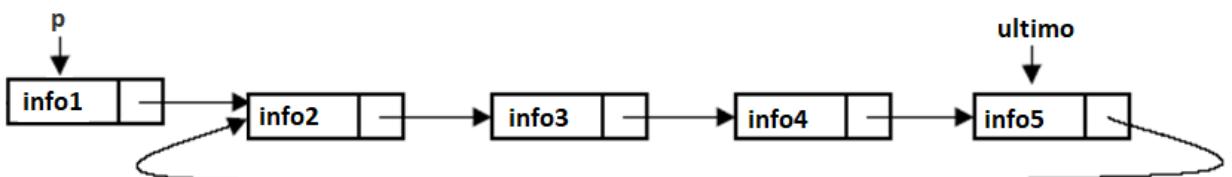
```
        ultimo = NULL;
```

```
    free(p);
```

```
}
```

```
return ultimo;
```

```
}
```



```

/* Exemplo de programa principal para uso da pilha*/
int main (){
PLista ultimo = NULL;
int op, val;
PLista laux;
ultimo = NULL;
do {
    printf("\nEscolha uma opcao\n");
    printf(" 1) Inserir um elemento na lista\n");
    printf(" 2) Retirar um elemento da lista\n");
    printf(" 3) Imprimir a lista\n");
    printf(" 4) Sair");
    printf("\nOpcao: ");
    scanf("%d",&op);
    switch (op) {
        case 1: printf("\nDigite um numero:");
            scanf("%d",&val);
            ultimo = Push(ultimo,val);
            break;
        case 2: if (ultimo==NULL)
                    printf("Pilha vazia!!!\n");
                else {
                    ultimo = Pop(ultimo,&val);
                    printf("\nElemento retirado da pilha: %d\n", val);
                }
            break;
    }
}

```

```

case 3: if(ultimo==NULL)
            printf("\nPilha vazia \n\n");
        else
            imprime(ultimo);
        break;

case 4: // libera espacos alocados antes de sair
        libera(ultimo);
        break;

default: if (op!=4)
            printf("Opcao errada. Digite novamente\n");
        break;
    }
}while(op!=4);

return 0;
}

```

Remoção em Lista Circular

- A remoção também pode ser feita do início, do final ou do meio. Já vimos a remoção a partir do início, que é a remoção em pilha vista anteriormente.

Exercício 2

- Faça as funções de inserção e remoção em Fila utilizando lista circular.

```
/* Insere na fila: inserção no fim da lista*/
```

```
PLista insere (PLista ultimo, int val){
```

```
    PLista novo = (PLista) malloc(sizeof(TLista));
```

```
    novo->info = val;
```

```
    if (ultimo == NULL)
```

```
        novo->prox = novo;
```

```
    else{
```

```
        novo->prox = ultimo->prox;
```

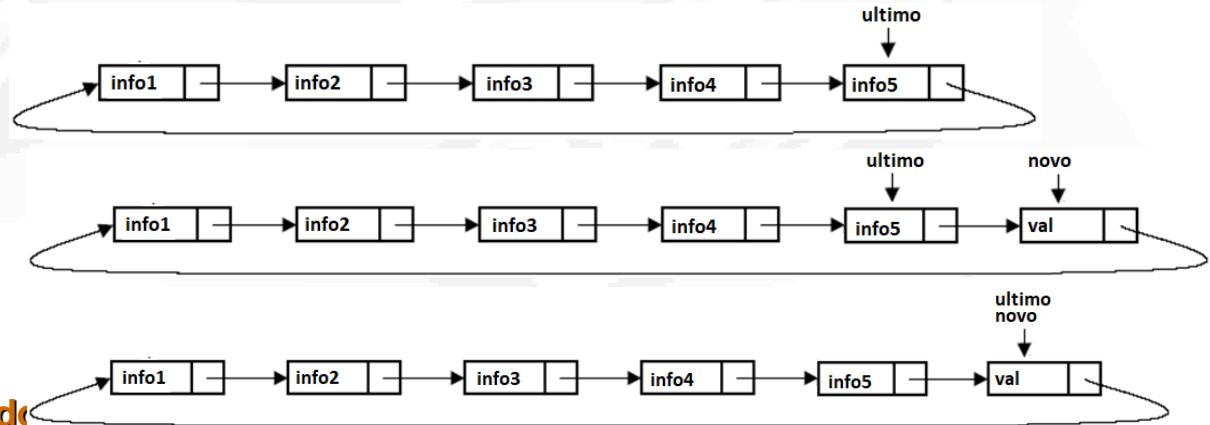
```
        ultimo->prox = novo;
```

```
}
```

```
    ultimo = novo; /* única alteração em relação à inserção já vista */
```

```
    return ultimo;
```

```
}
```



/ Remove da fila: remove do início da lista, ou seja, igual à função POP*/*

Exercício 3

- Escreva funções para efetuar cada uma das operações a seguir para listas circulares:
- ✓ Incluir um elemento no meio de uma lista (a posição a ser inserida deve ser passada como parâmetro da função)
 - ✓ Concatenar duas listas
 - ✓ Inverter uma lista
 - ✓ Eliminar o último elemento da lista
 - ✓ Eliminar o enésimo elemento da lista
 - ✓ Combinar duas listas ordenadas numa única lista ordenada
 - ✓ Colocar os elementos de uma lista em ordem crescente
 - ✓ Retornar o número de elementos da lista

Exercício 4

- Um grupo de soldados está rodeado por forças inimigas e não há chance de vitória sem a chegada de reforço. Como existe apenas um cavalo, os soldados entram num acordo para decidir quem irá fugir com o cavalo para tentar trazer ajuda. Eles formam um círculo e sorteiam num chapéu um número n e um nome. Começando pelo soldado cujo nome foi sorteado, é começado a contagem no sentido horário e quando esta contagem atingir o número n , este soldado é retirado do círculo. A contagem reinicia com o soldado seguinte ao que foi retirado e, novamente, quanto esta contagem atingir o número n , este outro soldado é retirado do círculo, recomeçando a contagem com o soldado seguinte, e assim por diante até que só reste um soldado no círculo. Este é o que deverá fugir no cavalo e buscar ajudar. Faça um programa que simule a formação dos soldados neste círculo (inclusão dos soldados no círculo) e a saída dos soldados na forma explicada até restar apenas um. Considere n e nome inicial do soldado pelo qual a contagem deve iniciar como fornecidos pelo usuário (para facilitar, considere que o usuário sempre entrará com um nome existente).

Exercício 5

➤ Considere um problema em que precisamos rotacionar um determinado conjunto de dados. Por exemplo, um conjunto com as letras “a b c d e”, se for rotacionado 2 vezes resulta em “c d e a b”, sendo que a 1a. rotação resulta em “b c d e a”. Considerando que não é permitido apenas trocar o valor de um nó para outro (é necessário realmente trocar as posições dos nós), escreva uma função para realizar essas operações.