Lista ligada por ponteiros

```c
#include <stdio.h>
#include <malloc.h>

typedef struct temp
{
    int data;
    struct temp *nextEl;
} list;
list *start, *newEl, *aux;
int listSize;

void insertElement(int pos, int d)
{
    if (pos < 1 || pos > listSize + 1)
    {
        printf("Posicao invalida");
        return;
    }

    newEl = (list *)malloc(sizeof(list));
    newEl->data = d;

    listSize++;

    if (pos == 1)
    {
        insertFirstEl();
        return;
    }

    moveAux(pos);

    newEl->nextEl = aux->nextEl;
    aux->nextEl = newEl;
}

void insertFirstEl()
{
    if (start == NULL)
        newEl->nextEl = NULL;
    else
        newEl->nextEl = start;

    start = newEl;
```

```
}

void removeElement(int pos)
{
    if (pos < 1 || pos > listSize)
        return;

    if (pos == 1)
    {
        removeFirstEl();
        return;
    }

    moveAux(pos);

    newEl = aux->nextEl;
    aux->nextEl = newEl->nextEl;
    free(newEl);
    listSize--;
}

void removeFirstEl()
{
    aux = start;
    start = start->nextEl;
    free(aux);
    listSize--;
}

void moveAux(int p)
{
    int i = 1;
    aux = start;

    while (i < p - 1)
    {
        aux = aux->nextEl;
        i++;
    }
}
```

```
void showElements()
{
    printf("Start");
    aux = start;

    int i = 0;
    while (i <= listSize)
    {
        printf("->%d", aux->data);
        aux = aux->nextEl;
        i++;
    }
}

int main()
{
    start = NULL;
    int op, d, p;

    do
    {
        printf("\n 1- Inserir elemento \t 2- Remover elemento \t 3- Listar \t 4-
Limpar Console \t 5- Sair");
        printf("\n");
        scanf("%d", &op);

        switch (op)
        {
        case 1:
            printf("\n valor:");
            scanf("%d", &d);
            printf("\n posicao:");
            scanf("%d", &p);
            insertElement(p, d);
            break;
        case 2:
            printf("\n posicao:");
            scanf("%d", &p);
            removeElement(p);
            break;
        case 3:
            showElements();
            break;
        case 4:
```

```
            system("cls");
            break;
        case 5:
            exit(0);
        default:
            printf("\n Operacao invalida!");
        }
    } while (op != 5);
}
```

Inserir elemento na lista vazia:

```
novo = (lista*)malloc(sizeof(lista)) // aloca espaço na memória
novo->dado = n // armazena valor
inicio = NULL
novo->prox = NULL   // o elemento novo é o primeiro da lista então o próximo é nulo
inicio = novo
```

-- caso for inserir na primeira posição

```
novo = (lista*)malloc(sizeof(lista))
novo->dado = n

novo->prox = inicio // o inicio não era nulo
inicio = novo       // o novo elemento se torna o primeiro e outro o segundo
```

Inserir elemento em uma posição da lista:

Obs: o elemento **B** é inserido na posição desejada, movendo o elemento **A** que estava na posição desejada para a sua posição seguinte.

C-**A**-Y -> C-**B**-A-Y

```
novo = (lista*)malloc(sizeof(lista))
novo->dado = n

aux = start
contador = 1
loop(contador <= posição - 1):    // o auxiliar para numa posição anterior a desejada
       aux = aux->prox
       contador++
novo->prox = aux->prox
aux->prox = novo
```

Remover o elemento na primeira posição:

```
aux = inicio
inicio = inicio->prox
free(aux)
```

Remover elemento em uma posição da lista:

```
aux = start
contador = 1
loop(contador <= posição - 1):
       aux = aux->prox
       contador++
novo = aux->prox
aux->prox = novo->prox
free(novo)
```