Manipulação de Lista
Professor Wladimir

Duplicar ímpares

Imagine que nós temos uma lista de números L. Por exemplo,

	0	1	2	3	4	5	6	7	8	9
L	8	4	1	9	17	12	26	13	7	14

A tarefa consiste em duplicar os elementos ímpares da lista obtendo a seguinte lista:

	0	1	2	3	4	5	6	7	8	9
L	8	4	2	9	34	12	26	26	14	14

Essa tarefa pode ser realizada com os seguintes passos:

1. Percorremos a lista da esquerda para a direira e sempre que encontramos um elemento ímpar, a gente multiplica esse elemento por 2.

```c
#include <stdio.h>

#define N 10
int L[N] = {8,4,1,9,17,12,26,13,7,14};

int main(){
    int i;
    for(i = 0; i < N; i++){
        if(L[i]%2==1) L[i] = L[i]*2;
    }
}
```

Trocando o maior elemento com o último

Imagine que nós temos uma lista de números L. Por exemplo,

	0	1	2	3	4	5	6	7	8	9
L	8	4	1	9	17	12	26	13	7	14

A tarefa consiste em trocar o maior elemento com o último elemento, obtendo a seguinte lista:

	0	1	2	3	4	5	6	7	8	9
L	8	4	1	9	17	12	14	13	7	26

Essa tarefa pode ser realizada com os seguintes passos:

1. Localize o maior elemento e a sua posição.
2. Troque o elemento da posição do maior com a última posição.

```c
#include <stdio.h>

#define N 10
int L[N] = {8,4,1,9,17,12,26,13,7,14};

int main(){
    int i, maior, posM;
    maior = L[0];
    posM = 0;
    for(i = 1; i < N; i++){
        if(L[i] > maior) {
            maior = L[i];
            posM = i;
        }
    }

    L[posM] = L[N-1];
    L[N-1] = maior;
}
```

Movendo o maior para a última posição

Imagine que nós temos uma lista de números L. Por exemplo,

	0	1	2	3	4	5	6	7	8	9
L	8	4	1	9	17	12	26	13	7	14

A tarefa consiste em mover o maior elemento para a última posição sem alterar a ordem dos demais elementos (deslocar todos os elementos à direita do maior), obtendo a seguinte lista:

	0	1	2	3	4	5	6	7	8	9
L	8	4	1	9	17	12	13	7	14	26

Essa tarefa pode ser realizada com os seguintes passos:

1. Localize o maior elemento e a sua posição.
2. Desloque todos os elementos à direita do maior 1 posição para a esquerda
3. Coloque o maior na última posição

```c
#include <stdio.h>

#define N 10
int L[N] = {8,4,1,9,17,12,26,13,7,14};

int main(){
    int i, maior, posM;
    maior = L[0];
    posM = 0;
    for(i = 1; i < N; i++){
        if(L[i] > maior) {
            maior = L[i];
            posM = i;
        }
    }

    for(int i = posM; i < N; i++){
        L[i] = L[i+1];
    }

    L[N-1] = maior;
}
```

Essa tarefa pode ser realizada com os seguintes passos:

1. Localize o maior elemento e a sua posição.
2. Criar um vetor auxiliar
3. Salvar os elementos do vetor na posição adequada.

```c
#include <stdio.h>

#define N 10
int L[N] = {8,4,1,9,17,12,26,13,7,14};

int main(){
    int i, maior, posM;
    maior = L[0];
    posM = 0;
    for(i = 1; i < N; i++){
        if(L[i] > maior) {
            maior = L[i];
            posM = i;
        }
    }

    //criacao do vetor auxiliar
    int A[N];

    for(int i = 0; i < N; i++){
        if(i<posM) A[i] = L[i];
        else if(i>posM) A[i-1] = L[i];
    }

    //Salvando o maior
    A[N-1] = maior;


}
```

Operação Varredura

A operação de varredura percorre a lista da esquerda para a direita, comparando elementos consecutivos. Sempre que um elemento da esquerda for maior que o elemento à sua direita, os dois serão trocados de posição. Perceba que, com esse procedimento, o maior elemento é ”empurrado”progressivamente para a última posição da lista, pois a cada comparação ele avança uma posição para a direita.

Imagine que nós temos uma lista de números L. Por exemplo,

	0	1	2	3	4	5	6	7	8	9
L	8	4	1	9	17	12	26	13	7	14

A operação de varredura realiza as seguintes trocas:

1. $8 \leftrightarrow 4$
2. $8 \leftrightarrow 1$
3. $17 \leftrightarrow 12$
4. $26 \leftrightarrow 13$
5. $26 \leftrightarrow 7$
6. $26 \leftrightarrow 14$

A operação de varredura obtém a seguinte lista:

	0	1	2	3	4	5	6	7	8	9
L	4	1	8	9	12	17	13	7	14	26

```c
#include <stdio.h>

#define N 10
int L[N] = {8,4,1,9,17,12,26,13,7,14};

int main(){
    int i;
    for(i = 1; i < N; i++){
        if(L[i-1] > L[i]) {
            int temp = L[i];
            L[i] = L[i-1];
            L[i-1] = temp;
        }
    }
}

```

```c
#include <stdio.h>

#define N 10
int L[N] = {8,4,1,9,17,12,26,13,7,14};

int main(){
    int i;
    for(i = 0; i < N-1; i++){
        if(L[i] > L[i+1]) {
            int temp = L[i];
            L[i] = L[i+1];
            L[i+1] = temp;
        }
    }
}

```

Colocando o maior na última posição e o menor na primeira posição

Imagine que nós temos uma lista de números L. Por exemplo,

	0	1	2	3	4	5	6	7	8	9
L	8	4	1	9	17	12	26	13	7	14

A tarefa consiste em colocar o maior elemento na última posição e o menor na primeira posição (sem manter a ordem dos demais elementos).

Essa tarefa pode ser realizada da seguinte maneira:

1. Realizando uma varredura para frente
2. E depois fazendo uma varredura para trás

```c
#include <stdio.h>

#define N 10
int L[N] = {8,4,1,9,17,12,26,13,7,14};

int main(){
    int i;
    //varredura para frente
    for(i = 0; i < N-1; i++){
        if(L[i] > L[i+1]) {
            int temp = L[i];
            L[i] = L[i+1];
            L[i+1] = temp;
        }
    }
    //varredura para trás
    for(i = N-2; i >=1; i--){
        if(L[i-1]  > L[i]) {
            int temp = L[i];
            L[i] = L[i-1];
            L[i-1] = temp;
        }
    }


}

```

Duplicar e arrumar

Imagine que nós temos uma lista de números L organizado em ordem crescente. Por exemplo,

	0	1	2	3	4	5	6	7	8	9
L	2	5	6	8	11	13	14	17	20	22

A tarefa consiste em dado uma posiçao k multiplique por 2 o elemento na posição k e mova o elemento para a posição correta de modo que a lista permaneça ordenada. Por exemplo, se $k = 3$ então o vetor arrumado será

	0	1	2	3	4	5	6	7	8	9
L	2	5	6	11	13	14	16	17	20	22

O vetor pode ser arrumado realizando uma varredura a apartir da posição k e a gente pode parar quando o elemento chegar na posição correta.

```c
#include <stdio.h>

#define N 10
int L[N] = {8,4,1,9,17,12,26,13,7,14};
int k = 3;

int main(){
    int i;

    L[k] = L[k]*2;
    //varredura para frente
    for(i = k; i < N-1; i++){
        if(L[i] > L[i+1]) {
            int temp = L[i];
            L[i] = L[i+1];
            L[i+1] = temp;
        }else break;
    }



}

```