Explicação do Código em C

Inclusão de Bibliotecas

```
#include <stdio.h>
#include <stdlib.h>
#include "interface.h"
```

- `#include <stdio.h>`: Inclui a biblioteca padrão de entrada e saída, que permite usar funções como `printf` e `scanf`.
- `#include <stdlib.h>`: Inclui a biblioteca que contém funções de alocação de memória e geração de números aleatórios (como `rand`).
- `#include "interface.h"`: Inclui um arquivo de cabeçalho que deve conter definições como `QTD_ELEM`, que indica a quantidade de elementos no vetor.

Função para Gerar Valores Aleatórios

```
void GeraValoresAleatorios(int *vetor) {
    for (int i = 1; i <= QTD_ELEM; i++) {
  vetor[i] = rand() % 100;  // Gera números aleatórios entre 0 e 99
    }
}
```

- **Descrição**: Esta função preenche o vetor com números aleatórios entre 0 e 99.
- **Parâmetro**: `int *vetor`: Um ponteiro para um vetor de inteiros.
- **Funcionamento**: Um loop que vai de 1 até `QTD_ELEM` gera um número aleatório e o armazena na posição correspondente do vetor.

Função para Imprimir o Vetor

```
void ImprimeVetor(int vetor[]) {
    for (int i = 1; i <= QTD_ELEM; i++) {
        printf("%d ", vetor[i]);
    }
    printf("\n");
}
```

- **Descrição**: Imprime todos os elementos do vetor na tela.

- **Parâmetro**: `int vetor[]`: Um vetor de inteiros.
- **Funcionamento**: Um loop percorre o vetor e imprime cada elemento seguido de um espaço.

Função Sift

```
void Sift(int i, int *vetor) {
    int esq = 2 * i;
    int dir = 2 * i + 1;
    int maior = i;

    if (esq <= QTD_ELEM && vetor[esq] > vetor[maior]) {
        maior = esq;
    }
    if (dir <= QTD_ELEM && vetor[dir] > vetor[maior]) {
        maior = dir;
    }
    if (maior != i) {
        int aux = vetor[i];
        vetor[i] = vetor[maior];
        vetor[maior] = aux;
        Sift(maior, vetor);  // Chamando recursivamente para
garantir propriedade estrutural
    }
}
```

- **Descrição**: Reorganiza um vetor para manter a propriedade do heap, garantindo que o maior elemento esteja na raiz.
- **Parâmetros**:
  - `int i`: O índice do elemento a ser reorganizado.
  - `int *vetor`: Um ponteiro para um vetor de inteiros.
- **Funcionamento**:
  - Calcula os índices dos filhos esquerdo e direito.
  - Compara o elemento atual com seus filhos e, se necessário, troca-os.
  - Chama a função `Sift` recursivamente para continuar o processo.

Função BuildHeap

```
void BuildHeap(int *vetor) {
    for (int i = QTD_ELEM / 2; i >= 1; i--) {
        Sift(i, vetor);
```

```
    }
}
```

- **Descrição**: Converte um vetor em um heap (estrutura de dados que permite acesso eficiente ao maior elemento).
- **Parâmetro**: `int *vetor`: Um ponteiro para um vetor de inteiros.
- **Funcionamento**: Um loop que começa do meio do vetor e chama a função `Sift` para cada elemento, garantindo que o vetor respeite as propriedades do heap.

Função para Determinar os Dois Maiores Elementos

```
void DeterminaDoisMaioresElementos(int *vetor) {
    if (QTD_ELEM < 2) {
        printf("O vetor deve ter pelo menos dois elementos.\n");
        return;
    }
    int max1 = vetor[1];
    int max2 = vetor[2] > vetor[3] ? vetor[2] : vetor[3];
    printf("Dois maiores elementos: %d e %d\n", max1, max2);
}
```

- **Descrição**: Encontra e imprime os dois maiores elementos do vetor.
- **Parâmetro**: `int *vetor`: Um ponteiro para um vetor de inteiros.
- **Funcionamento**:
    - Verifica se o vetor tem pelo menos dois elementos.
    - Atribui o maior elemento à variável `max1` e compara os próximos para encontrar o segundo maior.
    - Imprime os dois maiores elementos.

Função Principal (main)

```
int main() {
    int vetor[QTD_ELEM + 1];  // Vetor com 50 elementos,
considerando índice 1 como raiz

    GeraValoresAleatorios(vetor);
    printf("Vetor original:\n");
    ImprimeVetor(vetor);

    BuildHeap(vetor);
    printf("Vetor após BuildHeap:\n");
```

```
    ImprimeVetor(vetor);

    DeterminaDoisMaioresElementos(vetor);

    return 0;
}
```

- **Descrição**: Função principal que orquestra a execução do programa.
- **Funcionamento**:
    - Declara um vetor de inteiros de tamanho `QTD_ELEM + 1` (considerando que o índice 0 não será usado).
    - Chama `GeraValoresAleatorios` para preencher o vetor.
    - Imprime o vetor original.
    - Chama `BuildHeap` para transformar o vetor em um heap.
    - Imprime o vetor após a transformação.
    - Chama `DeterminaDoisMaioresElementos` para encontrar e exibir os dois maiores elementos.

Considerações Finais

Este programa é um exemplo de como gerar números aleatórios, manipular um vetor e implementar uma estrutura de dados chamada heap. A função `Sift` é essencial para manter a propriedade do heap durante a construção. A função `DeterminaDoisMaioresElementos` mostra como acessar elementos específicos do vetor após a organização.