

## Atividade 3 – AEDS III

Pedro Henrique Santos

### 1. Proposta

Implemente o método de ordenação interna *Heapsort* com base nos materiais disponibilizados nas aulas.

### 2. Código

Primeiro, temos a estrutura *Item*, que define o elemento a ser ordenado. Cada *Item* possui um valor *id*, que é o número a ser comparado para determinar a ordem.

```
struct Item {  
    int id;  
};
```

A função *troca* troca a posição de dois elementos do vetor, algo essencial para organizar os itens durante a ordenação.

```
void troca(Item& pai, Item& filho) {  
    Item aux = pai;  
    pai = filho;  
    filho = aux;  
}
```

A função *refaz* organiza o heap. Ela recebe um elemento chamado "pai" e compara seu valor com os "filhos". Se algum filho for maior, a função troca as posições do pai e do filho e repete o processo para o novo pai. Isso é feito até que o maior elemento chegue ao topo do heap, mantendo a ordem correta.

```
void refaz(vector<Item>& vetor, int pai, int tamanho) {  
    int filho = 2 * pai + 1;  
    while (filho < tamanho) {  
        if (filho + 1 < tamanho && vetor[filho].id < vetor[filho + 1].id)  
        {  
            filho++;  
        }  
        if (vetor[pai].id >= vetor[filho].id) {  
            break;  
        }  
        troca(vetor[pai], vetor[filho]);  
        pai = filho;  
        filho = 2 * pai + 1;  
    }  
}
```

Na função `heapsort`, o código começa organizando o vetor como um heap máximo. Para isso, ele executa a função `refaz` para cada elemento da metade inferior do vetor, assegurando que o maior número de cada parte esteja no topo do heap.

```
void heapsort(vector<Item>& vetor) {
    int tamanho = vetor.size();

    for (int i = tamanho / 2 - 1; i >= 0; i--) {
        refaz(vetor, i, tamanho);
    }

    for (int i = tamanho - 1; i > 0; i--) {
        troca(vetor[0], vetor[i]);
        refaz(vetor, 0, i);
    }
}
```

Depois de construir o heap, a função passa para a ordenação. A cada passo, o maior elemento (no topo do heap) é movido para o final do vetor e o tamanho do heap é reduzido em uma posição. Em seguida, `refaz` é chamada novamente para restaurar a propriedade do heap, mantendo o próximo maior elemento no topo. Isso se repete até que o vetor inteiro esteja em ordem crescente.

No final, temos a função `main`, que exibe o vetor original, chama o `heapsort` para ordenar e exibe o vetor ordenado.

```
int main() {
    SetConsoleOutputCP(65001);

    vector<Item> vetor = {{5}, {3}, {8}, {1}, {2}, {7}, {6}, {4}};

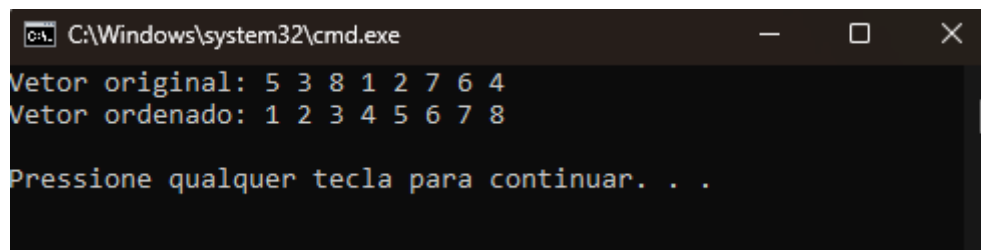
    cout << "Vetor original: ";
    for (const auto& item : vetor) {
        cout << item.id << " ";
    }
    cout << endl;

    heapsort(vetor);

    cout << "Vetor ordenado: ";
    for (const auto& item : vetor) {
        cout << item.id << " ";
    }
    cout << endl;

    return 0;
}
```

### 3. Resultado



```
C:\Windows\system32\cmd.exe
Vetor original: 5 3 8 1 2 7 6 4
Vetor ordenado: 1 2 3 4 5 6 7 8

Pressione qualquer tecla para continuar. . .
```

The image shows a Windows command prompt window with a dark background. The title bar at the top reads 'C:\Windows\system32\cmd.exe'. The command prompt displays two lines of text: 'Vetor original: 5 3 8 1 2 7 6 4' and 'Vetor ordenado: 1 2 3 4 5 6 7 8'. Below these, it prompts the user to 'Pressione qualquer tecla para continuar. . .' (Press any key to continue. . .). A vertical cursor is visible on the right side of the text area.