

Pedro Henrique Bezerra Cavalcante

Relatório Final

**Natal - RN
7 de junho de 2016**

Pedro Henrique Bezerra Cavalcante

Relatório Final

Relatório final para a disciplina de Introdução à Organização e Arquitetura de Computadores

Universidade Federal do Rio Grande do Norte

Orientador: Mônica Magalhães Pereira

Natal - RN
7 de junho de 2016

Sumário

	Sumário	2
1	INTRODUÇÃO	3
1.1	Daisy Chaining	3
1.2	Prioridade	3
1.3	Justiça	3
2	SOLUÇÃO E IMPLEMENTAÇÃO	4
3	ANÁLISE DOS RESULTADOS	8
4	CONCLUSÃO	9
5	MATERIAL UTILIZADO	10
6	DESCRIÇÃO E ORGANIZAÇÃO DO PROJETO E EXECUTAR	11

1 Introdução

1.1 Daisy Chaining

1.2 Prioridade

1.3 Justiça

2 Solução e Implementação

A solução encontrada para o resolvimento desse projeto foi encontrada, para Daisy Chaining, a execução dos periféricos de acordo com a entrada da solicitação, ou seja, pelo ordem que a requisição vai chegando ao barramento, ela vai ficar na fila e será executada conforme essa ordem.

Para prioridade, foi realizada uma ordenação utilizando o algoritmo de Ordenação por Seleção, de acordo com a prioridade armazenada na estrutura.

Foi desenvolvida uma lista encadeada com duas estruturas, mostradas à seguir:

```
1 typedef struct{
2     int tipo;
3     char dispositivo[50];
4     int prioridade;
5 }Perif;
6
7 typedef struct no{
8     Perif info;
9     struct no* prox;
10 }Aux_Perif;
```

A estrutura Aux_perif é formada pelo ponteiro para o próximo no e uma estrutura Perif. Essa última, é formada pelos dados do periférico, que é seu tipo (int), o nome do dispositivo (char) e a sua prioridade (int).

```
1 /**
2  * Arquivo main.cpp
3  */
4 #include <iostream>
5 #include <string.h>
6 #include <stdlib.h>
7
8 int main(){
9     int solic;
10    int cont = 0;
11    int aux = 0, prior = 0;
12    Aux_Perif* perifericos;
13    perifericos = criarLista();
14    std::cout << "Entre com o numero de solicitacoes: " <<
15        std::endl;
16    std::cin >> solic;
17    std::cout << "Perifericos:\n1. Impressora\n2. Mouse\n3.
18        Teclado\n4. Scanner\n5. Roteador\n6. Disco Rigido" <<
19        std::endl;
20    std::cout << "De acordo com os numeros e Perifericos
21        acima, insira a ordem de entrada das solicitacoes e
22        sua prioridade" << std::endl;
23    while (cont < solic){
```

```

19     Perif ordem;
20     std::cin >> aux >> prior;
21     switch (aux){
22         case 1:
23             //ordem.dispositivo = 'Impressora';
24             strcpy(ordem.dispositivo, "Impressora");
25             break;
26         case 2:
27             strcpy(ordem.dispositivo, "Mouse");
28             break;
29         case 3:
30             strcpy(ordem.dispositivo, "Teclado");
31             break;
32         case 4:
33             strcpy(ordem.dispositivo, "Scanner");
34             break;
35         case 5:
36             strcpy(ordem.dispositivo, "Roteador");
37             break;
38         case 6:
39             strcpy(ordem.dispositivo, "Disco Rigido");
40             break;
41         default:
42             std::cout << "Error" << std::endl;
43             break;
44     }
45
46     ordem.tipo = aux;
47     ordem.prioridade = prior;
48     cont++;
49     inserirFinal(&perifericos, ordem);
50     //delete[] ordem;
51
52 }
53 std::cout << "Daisy Chaining utilizando metodo de ordem a
54 chegada ou entrada." << std::endl;
55 imprimirLista(&perifericos);
56 std::cout << "METODO POR PRIORIDADE: " << std::endl;
57 ordena_AuxPerif(&perifericos);
58 imprimirLista(&perifericos);
59 return 0;
60 }
61 /**
62 * Funcoes utilizadas
63 */
64 typedef struct{
65     int tipo;
66     char dispositivo[50];
67     int prioridade;

```

```

67 }Perif;
68
69 typedef struct no{
70     Perif info;
71     struct no* prox;
72 }Aux_Perif;
73
74 Aux_Perif* criarLista(){
75     return NULL;
76 }
77 void selectionsort_AuxPerif(Aux_Perif **vetor, int tamanho) {
78     int i, j;
79     for (i = 0; i < tamanho - 1; i++) {
80         int menor = i;
81         for (j = i + 1; j < tamanho; j++) {
82             if (vetor[menor]->info.prioridade > vetor[j]->
                info.prioridade) menor = j;
83         }
84         if (menor != i) {
85             Aux_Perif *temp = vetor[menor];
86             vetor[menor] = vetor[i];
87             vetor[i] = temp;
88         }
89     }
90 }
91
92 void ordena_AuxPerif(Aux_Perif **perif) {
93     // 1. Se a lista esta vazia, entao nem faz nada.
94     if (perif == NULL) return;
95
96     // 2. Descobre o tamanho da lista.
97     int tamanho = 0;
98     Aux_Perif *v;
99     for (v = *perif; v != NULL; v = v->prox) {
100         tamanho++;
101     }
102
103     // 3. Monta um vetor com os elementos da lista.
104     Aux_Perif **vetor = (Aux_Perif **) malloc(sizeof(
        Aux_Perif *) * tamanho);
105     int i = 0;
106     for (v = *perif; v != NULL; i++) {
107         vetor[i] = v;
108         v = v->prox;
109     }
110
111     // 4. Ordena o vetor.
112     selectionsort_AuxPerif(vetor, tamanho);
113

```

```

114     // 5. Corrige os ponteiros de acordo com a nova ordenacao
115     .
116     for (i = 0; i < tamanho - 1; i++) {
117         vetor[i]->prox = vetor[i + 1];
118     }
119     vetor[i]->prox = NULL;
120
121     // 6. Corrige o ponteiro para o inicio da lista.
122     *perif = vetor[0];
123     //void imprimirLista(*perif);
124
125     // 7. Deleta o vetor auxiliar.
126     free(vetor);
127 }
128
129 void imprimirLista(Aux_Perif** lista){
130     Aux_Perif* aux = *lista;
131     while (aux != NULL){
132         std::cout << "DISPOSITIVO: ";
133         std::cout << aux->info.dispositivo << std::endl;
134         std::cout << "PRIORIDADE: ";
135         std::cout << aux->info.prioridade << std::endl;
136         aux = aux->prox;
137     }
138 }
139
140 void inserirFinal(Aux_Perif** perifericos, Perif barramento){
141     Aux_Perif* novo = (Aux_Perif*)new Aux_Perif;
142     novo->info = barramento;
143     Aux_Perif* aux = *perifericos;
144     Aux_Perif* anterior = NULL;
145     if (*perifericos == NULL){
146         *perifericos = novo;
147     }else{
148         while(aux != NULL){
149             anterior = aux;
150             aux = aux->prox;
151         }
152         anterior->prox = novo;
153     }
154 }

```


3 Análise dos Resultados

4 Conclusão

5 Material Utilizado

6 Descrição e Organização do Projeto e Executar