FILA DE PIORIDADES

Trabalho Avaliativo 2 - Algoritmo e Estrutura de Dados 1
Alunos: Leonardo Caparica e Milena Bernardi

Simulação Controle de Tráfego Aéreo

Processo de Simulação:

1. Inicialização:
   - A simulação começa com a inicialização de uma fila de entrada contendo até 50 voos. Cada voo é criado com um número único, uma prioridade aleatória entre 1 e 3, uma indicação se é para pouso ou decolagem, e um tempo fixo de execução de 2 unidades de tempo. Voos com prioridade 3 são para decolagem, enquanto os demais são para pouso.

2. Estruturas de Dados:
   - Fila de Entrada: Contém os voos que solicitam pouso ou decolagem.
   - Fila de Prioridades: Uma fila de prioridade que organiza os voos de acordo com sua prioridade e número, com maior prioridade para números menores.
   - Pistas: Um vetor de três pistas onde os voos são processados.

3. Simulação do Controle de Tráfego:
   - A cada unidade de tempo:
     - Até 4 voos são movidos da fila de entrada para a fila de prioridades.
     - O tempo de execução dos voos nas pistas é decrementado. Se o tempo de execução de um voo atinge zero, a pista fica livre.
     - Voos da fila de prioridades são alocados às pistas livres.
   - A simulação continua até que todas as pistas estejam livres e não haja mais voos nas filas.

4. Relatórios e Exibição:
   - A cada unidade de tempo, o estado atual das pistas e da fila de prioridades é exibido em uma tabela formatada.
   - A simulação inclui um menu interativo que permite ao usuário iniciar a simulação, visualizar a fila de entrada, ou sair do programa.

Detalhes do Sistema:

- Pistas: Três pistas são usadas para processar voos, e cada pista pode estar livre ou ocupada com um voo.

- Solicitações: Os voos podem ser para pouso ou decolagem. Voos de prioridade 3 são decolagens, e os demais são pousos.

- Prioridades: Voos são atribuídos uma prioridade de 1 a 3, com 1 sendo a mais alta. A fila de prioridades organiza os voos para garantir que os de maior prioridade (e, em caso de empate, os com menor número) sejam processados primeiro.
- Unidades de Tempo: A cada unidade de tempo, voos são movidos entre filas, processados nas pistas, e o estado do sistema é atualizado e exibido.

Esse modelo permite a visualização clara e interativa de como um sistema de controle de tráfego aéreo pode gerenciar múltiplas solicitações de voos, assegurando que as prioridades sejam respeitadas e as pistas sejam utilizadas de maneira eficiente.

Agora vamos para o código:

```
1   #include <iostream>
2   #include <queue>
3   #include <vector>
4   #include <string>
5   #include <iomanip> // Para a formatação da Tabela
6   #include <cstdlib> // Para a função rand()
7   #include <ctime>   // Para a função time()
8   #include <unistd.h>
9   #define QNTVOOS 50
```

Bibliotecas utilizadas no código, e utilizamos o #define para definir a quantidade de voos desejada.

```
13  struct Voo
14  {
15      int numero;
16      int prioridade;
17      bool manobra;
18      int tempoExecucao;
19
20      bool operator<(const Voo &outro) const
21      {
22          if (prioridade != outro.prioridade)
23          {
24              return prioridade > outro.prioridade;
25          }
26          else
27          {
28              return numero > outro.numero;
29          }
30      }
31  };
32
33  // Fila de prioridades para pousos e decolagens
34  priority_queue<Voo> filaPrioridades;
35
36  // Fila de entrada para voos que solicitam pouso ou decolagem
37  queue<Voo> filaEntrada;
38  queue<Voo> filaAmostra;
39
40  // Vetor para armazenar as pistas
41  vector<Voo> pistas(3);
```

A “struct” definida como “Voo”, contendo inteiros e booleano para ser feito a prioridade, em seguida a associação do comando pela biblioteca da fila (queue), as filas de prioridade para os voos que solicitam pouso ou decolagem. Vetor para definir as pistas (3).

```
43    // Prototipagem das Funções
44    void inicializarFilaEntrada();
45    void simularControleTrafego();
46    Voo inicializaVoo(int num);
47    void mostraVoo(queue<Voo> filaEntrada);
48    void mostraControleTrafegoTabela(int tempo);
49    void mostrarFilaPrioridades();
50    void menu();
51
52    int main (){
53        srand(time(NULL));
54        inicializarFilaEntrada();
55        menu();
56        return 0;
57    }
```

Funções utilizadas dentro do código, juntamente com a int main.

```
59  void menu()
60  {
61      int escolha;
62      do
63      {
64          cout << "===== Menu =====" << endl;
65          cout << "1. Iniciar simulacao de controle de trafego aereo" << endl;
66          cout << "2. Mostrar voos na fila de entrada" << endl;
67          cout << "3. Sair" << endl;
68          cout << "Escolha uma opcao: ";
69          cin >> escolha;
70
71          switch (escolha)
72          {
73          case 1:
74              cout << "Voos na fila de entrada:" << endl;
75              mostraVoo(filaAmostra);
76              simularControleTrafego();
77              break;
78          case 2:
79              cout << "Voos na fila de entrada:" << endl;
80              mostraVoo(filaAmostra);
81              break;
82          case 3:
83              cout << "Saindo..." << endl;
84              break;
85          default:
86              cout << "Opcao invalida! Escolha novamente." << endl;
87              break;
88          }
89      } while (escolha != 3);
90      sleep(3);
91  }
```

```
===== Menu =====
1. Iniciar simulacao de controle de trafego aereo
2. Mostrar voos na fila de entrada
3. Sair
Escolha uma opcao:
```

A implementação contém um menu, que faz com que a saída no terminal seja mais dinâmica e organizada.

```
 93    Voo inicializaVoo(int num)
 94    {
 95        Voo voo;
 96        voo.numero = num;
 97        voo.prioridade = 1 + rand() % 3;
 98        voo.manobra = (voo.prioridade != 3);
 99        voo.tempoExecucao = 2;
100        return voo;
101    }
```

A função inicializaVoo cria e retorna uma estrutura Voo inicializada com um número fornecido, uma prioridade aleatória entre 1 e 3, um valor booleano para manobra (verdadeiro se a prioridade não for 3) e um tempo de execução fixo de 2.

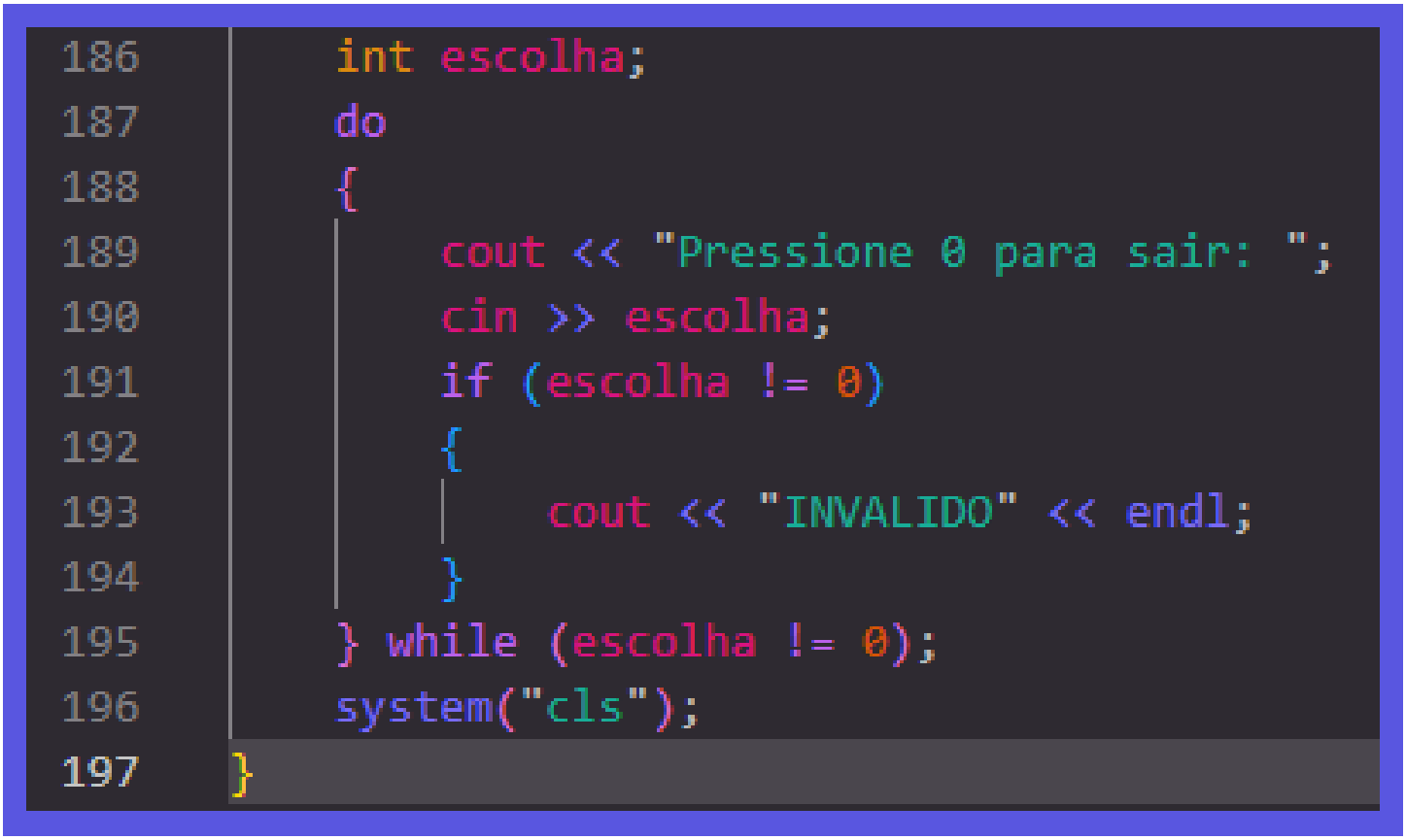
```
103   void mostraVoo(queue<Voo> filaEntrada)
104   {
105       system("cls");
106       while (!filaEntrada.empty())
107       {
108           Voo v = filaEntrada.front();
109           filaEntrada.pop();
110           cout << "Numero: " << v.numero << endl
111                << "Prioridade: " << v.prioridade << endl
112                << "Manobra: " << (v.manobra ? "Pouso" : "Decolagem") << endl
113                << endl;
114       }
115       int escolha;
116       do
117       {
118           cout << "Pressione 0 para continuar: ";
119           cin >> escolha;
120       } while (escolha != 0);
121       system("cls");
122   }
```

```
Numero: 1
Prioridade: 2
Manobra: Pouso

Numero: 2
Prioridade: 2
Manobra: Pouso

Numero: 3
Prioridade: 2
Manobra: Pouso

Numero: 4
Prioridade: 3
Manobra: Decolagem

Numero: 5
Prioridade: 3
Manobra: Decolagem

Numero: 6
Prioridade: 1
Manobra: Pouso

Numero: 7
Prioridade: 2
Manobra: Pouso

Numero: 8
Prioridade: 3
Manobra: Decolagem

Numero: 9
Prioridade: 2
Manobra: Pouso

Numero: 10
Prioridade: 3
Manobra: Decolagem
```

Caso escolha a opção 2 na saída do terminal, ele te retornará a void mostraVoo, onde será possível observar o número do voo, sua prioridade e manobra. (A saída contém até o número 50).

```
void simularControleTrafego()
{
    int tempo = 0;
    bool todasLivres = false;

    while (!todasLivres || !filaPrioridades.empty() || !filaEntrada.empty())
    {
        todasLivres = true;

        // A cada instante de tempo, 4 voos saem da fila de entrada e entram na fila de prioridades
        for (int i = 0; i < 4 && !filaEntrada.empty(); i++)
        {
            Voo voo = filaEntrada.front();
            filaEntrada.pop();
            filaPrioridades.push(voo);
        }

        // Atualizar tempo de execução dos voos nas pistas
        for (int i = 0; i < 3; i++)
        {
            if (pistas[i].numero != 0)
            {
                todasLivres = false; // Ainda há pelo menos uma pista ocupada
                pistas[i].tempoExecucao--;
                if (pistas[i].tempoExecucao == 0)
                {
                    // Manobra concluída, pista fica livre
                    pistas[i] = {0, 0, false, 0};
                }
            }
        }

        // Alocar novos voos às pistas livres
        for (int i = 0; i < 3; i++)
        {
            if (pistas[i].numero == 0 && !filaPrioridades.empty())
            {
                Voo voo = filaPrioridades.top();
                filaPrioridades.pop();
                pistas[i] = voo;
                todasLivres = false; // Uma pista foi alocada
            }
        }

        if (todasLivres)
        {
            break; // Se todas as pistas estiverem livres, encerra a simulação
        }

        mostraControleTrafegoTabela(tempo);

        tempo++;
    }
```

A função simularControleTrafego simula a gestão do tráfego aéreo. Inicializa o tempo e uma variável de controle das pistas. Em um loop, enquanto houver voos para processar ou pistas ocupadas, quatro voos são movidos da fila de entrada para a fila de prioridades a cada instante de tempo. As pistas são verificadas e o tempo de execução dos voos nelas é decrementado. Se o tempo de execução de um voo se esgota, a pista é liberada. Em seguida, as pistas livres recebem novos voos da fila de prioridades. Se todas as pistas estão livres e não há mais voos para processar, a simulação termina. A função também atualiza e mostra o estado do controle de tráfego a cada instante de tempo.



```
186     int escolha;
187     do
188     {
189         cout << "Pressione 0 para sair: ";
190         cin >> escolha;
191         if (escolha != 0)
192         {
193             cout << "INVALIDO" << endl;
194         }
195     } while (escolha != 0);
196     system("cls");
197 }
```

```
===== Menu =====
1. Iniciar simulacao de controle de trafego aereo
2. Mostrar voos na fila de entrada
3. Sair
Escolha uma opcao: 0
Opcao invalida! Escolha novamente.
===== Menu =====
1. Iniciar simulacao de controle de trafego aereo
2. Mostrar voos na fila de entrada
3. Sair
Escolha uma opcao:
```

Se a opção for diferente de algum número que não houver, ele retorna “Opção inválida! Escolha novamente.” e retorna para onde estava para escolher o número correto.

```
210  void mostraControleTrafegoTabela(int tempo)
211  {
212      vector<string> linha(4);
213      cout << endl;
214      linha[0] = "Tempo: " + to_string(tempo);
215      for (int i = 0; i < 3; i++)
216      {
217          if (pistas[i].numero != 0)
218          {
219              linha[i + 1] = "Voo: " + to_string(pistas[i].numero) + " | " +
220                             (pistas[i].manobra ? "P" : "D") + "  | Priori: " +
221                             to_string(pistas[i].prioridade) + "    ";
222          }
223          else
224          {
225              linha[i + 1] = "        Pista Livre         ";
226          }
227      }
228
229      cout << linha[0] << endl;
230      cout << "-------------------------------------------------------------------------------------------------------" << endl;
231      cout << "|          Pista 1          |          Pista 2          |          Pista 3          |" << endl;
232      cout << "|" << setw(20) << linha[1] << "|" << setw(20) << linha[2] << "|" << setw(20) << linha[3] << "|" << endl;
233      cout << "_______________________________________________________________________________________________________" << endl;
234      cout << endl;
235      // Exibir voos na fila de prioridades em espera
236      cout << "Fila de Prioridades em Espera: ";
237      mostrarFilaPrioridades();
238      cout << endl
239          << endl;
240
241  }
```

```
Tempo: 0
-------------------------------------------------------------------------------------------------------------
|          Pista 1          |          Pista 2          |          Pista 3          |
|Voo: 1 | P  | Priori: 2    |Voo: 2 | D  | Priori: 3    |Voo: 3 | D  | Priori: 3    |
-------------------------------------------------------------------------------------------------------------

Fila de Prioridades em Espera: 4


Tempo: 1
-------------------------------------------------------------------------------------------------------------
|          Pista 1          |          Pista 2          |          Pista 3          |
|Voo: 1 | P  | Priori: 2    |Voo: 2 | D  | Priori: 3    |Voo: 3 | D  | Priori: 3    |
-------------------------------------------------------------------------------------------------------------

Fila de Prioridades em Espera: 5 6 8 4 7


Tempo: 2
-------------------------------------------------------------------------------------------------------------
|          Pista 1          |          Pista 2          |          Pista 3          |
|Voo: 5 | P  | Priori: 1    |Voo: 12 | P  | Priori: 1    |Voo: 6 | P  | Priori: 2    |
-------------------------------------------------------------------------------------------------------------

Fila de Prioridades em Espera: 8 9 11 4 7 10


Tempo: 3
-------------------------------------------------------------------------------------------------------------
|          Pista 1          |          Pista 2          |          Pista 3          |
|Voo: 5 | P  | Priori: 1    |Voo: 12 | P  | Priori: 1    |Voo: 6 | P  | Priori: 2    |
-------------------------------------------------------------------------------------------------------------

Fila de Prioridades em Espera: 14 8 9 11 15 16 4 7 10 13


Tempo: 4
-------------------------------------------------------------------------------------------------------------
|          Pista 1          |          Pista 2          |          Pista 3          |
|Voo: 14 | P  | Priori: 1    |Voo: 18 | P  | Priori: 1    |Voo: 19 | P  | Priori: 1    |
-------------------------------------------------------------------------------------------------------------
```

Para a saída da opção número 1, ela começa mostrando a mesma coisa que a opção 2, e em seguida pede para digitar “0” para continuar. Logo que é continuado ele mostra essa “tabela” com o tempo na parte superior, abaixo as pistas, uma ao lado da outra e um pouco mais abaixo o número do voo, manobra e prioridade, finalizando com a fila de prioridade em espera, podendo assim ver quais serão os próximos voos de acordo com a sua prioridade.

Código completo:

```cpp
#include <iostream>
#include <queue>
#include <vector>
#include <string>
#include <iomanip> // Para a formatação da Tabela
#include <cstdlib> // Para a função rand()
#include <ctime>   // Para a função time()
#include <unistd.h>
#define QNTVOOS 50

using namespace std;

struct Voo
{
    int numero;
    int prioridade;
    bool manobra;
    int tempoExecucao;

    bool operator<(const Voo &outro) const
    {
        if (prioridade != outro.prioridade)
        {
            return prioridade > outro.prioridade;
        }
        else
        {
            return numero > outro.numero;
        }
    }
};

// Fila de prioridades para pousos e decolagens
priority_queue<Voo> filaPrioridades;

// Fila de entrada para voos que solicitam pouso ou decolagem
queue<Voo> filaEntrada;
queue<Voo> filaAmostra;

// Vetor para armazenar as pistas
vector<Voo> pistas(3);

// Prototipagem das Funções
void inicializarFilaEntrada();
void simularControleTrafego();
Voo inicializaVoo(int num);
void mostraVoo(queue<Voo> filaEntrada);
void mostraControleTrafegoTabela(int tempo);
void mostrarFilaPrioridades();
```

```
void mostrarFilaPrioridades();
void menu();

int main (){
    srand(time(NULL));
    inicializarFilaEntrada();
    menu();
    return 0;
}

void menu()
{
    int escolha;
    do
    {
        cout << "===== Menu =====" << endl;
        cout << "1. Iniciar simulacao de controle de trafego aereo" << endl;
        cout << "2. Mostrar voos na fila de entrada" << endl;
        cout << "3. Sair" << endl;
        cout << "Escolha uma opcao: ";
        cin >> escolha;

        switch (escolha)
        {
        case 1:
            cout << "Voos na fila de entrada:" << endl;
            mostraVoo(filaAmostra);
            simularControleTrafego();
            break;
        case 2:
            cout << "Voos na fila de entrada:" << endl;
            mostraVoo(filaAmostra);
            break;
        case 3:
            cout << "Saindo..." << endl;
            break;
        default:
            cout << "Opcao invalida! Escolha novamente." << endl;
            break;
        }
    } while (escolha != 3);
    sleep(3);
}

Voo inicializaVoo(int num)
{
    Voo voo;
    voo.numero = num;
```

```
    voo.prioridade = 1 + rand() % 3;
    voo.manobra = (voo.prioridade != 3);
    voo.tempoExecucao = 2;
    return voo;
}

void mostraVoo(queue<Voo> filaEntrada)
{
    system("cls");
    while (!filaEntrada.empty())
    {
        Voo v = filaEntrada.front();
        filaEntrada.pop();
        cout << "Numero: " << v.numero << endl
             << "Prioridade: " << v.prioridade << endl
             << "Manobra: " << (v.manobra ? "Pouso" : "Decolagem") << endl
             << endl;
    }
    int escolha;
    do
    {
        cout << "Pressione 0 para continuar: ";
        cin >> escolha;
    } while (escolha != 0);
    system("cls");
}

void inicializarFilaEntrada()
{
    for (int i = 0; i < QNTVOOS; i++)
    {
        filaEntrada.push(inicializaVoo(i + 1));
    }
    filaAmostra = filaEntrada;
}

void simularControleTrafego()
{
    int tempo = 0;
    bool todasLivres = false;

    while (!todasLivres || !filaPrioridades.empty() || !filaEntrada.empty())
    {
        todasLivres = true;

        // A cada instante de tempo, 4 voos saem da fila de entrada e entram na fila de prioridades
        for (int i = 0; i < 4 && !filaEntrada.empty(); i++)
```

```
144         {
145             Voo voo = filaEntrada.front();
146             filaEntrada.pop();
147             filaPrioridades.push(voo);
148         }
149 
150         // Atualizar tempo de execução dos voos nas pistas
151         for (int i = 0; i < 3; i++)
152         {
153             if (pistas[i].numero != 0)
154             {
155                 todasLivres = false; // Ainda há pelo menos uma pista ocupada
156                 pistas[i].tempoExecucao--;
157                 if (pistas[i].tempoExecucao == 0)
158                 {
159                     // Manobra concluída, pista fica livre
160                     pistas[i] = {0, 0, false, 0};
161                 }
162             }
163         }
164 
165         // Alocar novos voos às pistas livres
166         for (int i = 0; i < 3; i++)
167         {
168             if (pistas[i].numero == 0 && !filaPrioridades.empty())
169             {
170                 Voo voo = filaPrioridades.top();
171                 filaPrioridades.pop();
172                 pistas[i] = voo;
173                 todasLivres = false; // Uma pista foi alocada
174             }
175         }
176 
177         if (todasLivres)
178         {
179             break; // Se todas as pistas estiverem livres, encerra a simulação
180         }
181 
182         mostraControleTrafegoTabela(tempo);
183 
184         tempo++;
185     }
186     int escolha;
187     do
188     {
```

```
        cout << "Pressione 0 para sair: ";
        cin >> escolha;
        if (escolha != 0)
        {
            cout << "INVALIDO" << endl;
        }
    } while (escolha != 0);
    system("cls");
}

void mostrarFilaPrioridades()
{
    priority_queue<Voo> temp = filaPrioridades;
    while (!temp.empty())
    {
        Voo v = temp.top();
        temp.pop();
        cout << v.numero << " ";
    }
}

void mostraControleTrafegoTabela(int tempo)
{
    vector<string> linha(4);
    cout << endl;
    linha[0] = "Tempo: " + to_string(tempo);
    for (int i = 0; i < 3; i++)
    {
        if (pistas[i].numero != 0)
        {
            linha[i + 1] = "Voo: " + to_string(pistas[i].numero) + " | " +
                           (pistas[i].manobra ? "P" : "D") + "  | Priori: " +
                           to_string(pistas[i].prioridade) + "    ";
        }
        else
        {
            linha[i + 1] = "        Pista Livre         ";
        }
    }

    cout << linha[0] << endl;
    cout << "-------------------------------------------------------------------------------------------------------------" << endl;
    cout << "|           Pista 1          |          Pista 2           |          Pista 3           |" << endl;
    cout << "|" << setw(20) << linha[1] << "|" << setw(20) << linha[2] << "|" << setw(20) << linha[3] << "|" << endl;
    cout << "_____________________________________________________________________________________________________________" << endl;
    cout << endl;
    // Exibir voos na fila de prioridades em espera
```

```
    // Exibir voos na fila de prioridades em espera
    cout << "Fila de Prioridades em Espera: ";
    mostrarFilaPrioridades();
    cout << endl
         << endl;

}
```