Mateus Lopes de Oliveira
Yan Gabriel Reis Oliveira

O algoritmo considera um sistema de controle de pouso e decolagem de aviões utilizando o algoritmo de filas de prioridade para determinar as ações de cada avião.
Nosso sistema possui 3 pistas que podem ser usadas por 2 tipos diferentes de ação por avião: Pouso e Decolagem.
O pouso pode ter prioridade 1 (alta) e 2 (média), já a decolagem tem sempre prioridade 3 (baixa).
Cada pista pode ter somente um avião em cada instante de tempo, e cada avião leva 2 instantes de tempo para terminar o processo.
A cada instante de tempo, 3 aviões saem da lista de chegada para entrar na fila de prioridades, e depois checa se as pistas estão vazias. Se for o caso, o algoritmo coloca os 3 primeiros aviões da lista de prioridade em suas pistas correspondentes.
Resumindo: Aviões chegam de unidade de tempo em unidade de tempo, mas levam 2 unidades de tempo para realizarem pouso/decolagem.
Tanto a fila de prioridade quanto o estado de cada uma das pistas é atualizado de instante a instante no console, sendo sempre reescrita a fim de manter legibilidade a cada unidade de tempo do sistema.

```
Avioes esperando instrucoes:
|AVIAO: 100 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 101 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 102 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 103 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 104 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 105 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 106 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 107 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 108 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 109 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 110 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 111 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 112 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 113 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 114 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 115 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 116 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 117 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 118 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 119 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 120 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 121 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 122 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 123 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 124 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 125 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 126 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 127 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 128 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 129 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 130 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 131 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 132 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 133 |PRIORIDADE: 3 |REQUISITA: D|
```

```
_____________________________________________
            Unidade de Tempo: 3

FILA DE AVIOES ESPERANDO:
|AVIAO: 105 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 108 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 109 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 110 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 111 |PRIORIDADE: 1 |REQUISITA: P|
|AVIAO: 106 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 107 |PRIORIDADE: 2 |REQUISITA: P|
|AVIAO: 101 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 104 |PRIORIDADE: 3 |REQUISITA: D|

PISTA 1 sendo usada pelo aviao 105 <<<
PISTA 2 sendo usada pelo aviao 108 <<<
PISTA 3 sendo usada pelo aviao 109 <<<
```

```
_____________________________________________
            Unidade de Tempo: 33

FILA DE AVIOES ESPERANDO:
|AVIAO: 144 |PRIORIDADE: 3 |REQUISITA: D|
|AVIAO: 146 |PRIORIDADE: 3 |REQUISITA: D|

PISTA 1 sendo usada pelo aviao 144 >>>
PISTA 2 sendo usada pelo aviao 146 >>>

SEM AVIOES NA FILA - FILA VAZIA


FILA DE AVIOES ESPERANDO:
SEM AVIOES ESPERANDO PARA POUSAR/DECOLAR
```

```
_____________________________________________
            Unidade de Tempo: 35

FILA DE AVIOES ESPERANDO:
SEM AVIOES ESPERANDO PARA POUSAR/DECOLAR

TODOS OS AVIOES POUSARAM/DECOLARAM! Sem trafego aereo no momento
```

```
#include <iostream>
#include <chrono>
#include <ctime>
#include <thread>
using namespace std;
```

```
using namespace std::this_thread;
using namespace std::chrono_literals;
#include "FilaPrioridadesLEAviao.h"

void imprimePistaDePouso(int,char,int);

int main(void)
{
    node *chegadaDeAvioes, *filaPrioridade;
     int tempo, aviao, idAviao, idRand,prioridade, i, aviaoNaPista[3],
quantAvioes = 0;
    char tipo, tipoNaPista[3], tipoRand;;
    bool isVazia = false, isPistaVazia = false;

    srand(time(NULL));

    chegadaDeAvioes = inicializaFP(chegadaDeAvioes);
    filaPrioridade = inicializaFP(filaPrioridade);

    for(int i = 0; i < 50; i++)
    {
        idRand = (rand() % 3)+1;
        if(idRand == 1 || idRand == 2)
            tipoRand = 'P';
        else
            tipoRand = 'D';
           chegadaDeAvioes = insereFilaChegada(chegadaDeAvioes, 100+i,
tipoRand, idRand);
    }

    cout << "Avioes esperando instrucoes:\n";
    exibe(chegadaDeAvioes);
    sleep_for(1s);
    for(tempo = 1; !isVazia || (tempo%2); tempo++)
    {
        system("clear");
        cout << "\n_____________________________________________" << endl;
          cout << "             " << "Unidade de Tempo: " << tempo << "
\n" << endl;
        sleep_for(0.25s);
        for (aviao = 1; aviao <= 4; aviao++){
            if (!verificaSeVazia(chegadaDeAvioes)){
```

```
                    chegadaDeAvioes = removeFP(chegadaDeAvioes, &idAviao,
&tipo, &prioridade);
                        filaPrioridade = insereFP(filaPrioridade, idAviao,
tipo, prioridade);
            }
        }
        cout << "FILA DE AVIOES ESPERANDO:\n";
        if(!verificaSeVazia(filaPrioridade))
            exibe(filaPrioridade);
        else
            cout << "SEM AVIOES ESPERANDO PARA POUSAR/DECOLAR\n";

        if ((tempo%2) && (!isVazia)){
            cout << endl;
            for (i = 1; i <= 3; i++){
                if (filaPrioridade != NULL){
                    filaPrioridade = removeFP(filaPrioridade, &idAviao,
&tipo, &prioridade);
                    aviaoNaPista[i-1] = idAviao;
                    tipoNaPista[i-1] = tipo;
                    quantAvioes++;
                    imprimePistaDePouso(i, tipo, idAviao);
                }
            }
            if (!verificaSeVazia(filaPrioridade)){
                sleep_for(0.25s);
                cout << endl << "FILA DE AVIOES ESPERANDO:\n";
                exibe(filaPrioridade);
            }
            else
            {
                sleep_for(0.25s);
                cout << endl << "SEM AVIOES NA FILA - FILA VAZIA\n";
                sleep_for(0.25s);
                isPistaVazia = true;
                sleep_for(0.25s);
                cout << endl << "\nFILA DE AVIOES ESPERANDO:\n";
                cout << "SEM AVIOES ESPERANDO PARA POUSAR/DECOLAR\n";
                sleep_for(0.25s);
            }
        }
        else if(isPistaVazia && (!isVazia)){
            sleep_for(0.25s);
```

```
                                cout << endl << "-------------PISTAS
OCUPADAS--------------\n" << endl;
            sleep_for(0.25s);
            for(i = 1; i <= quantAvioes; i++)
            {
                                imprimePistaDePouso(i, tipoNaPista[i-1],
aviaoNaPista[i-1]);
            }
            quantAvioes = 0;
            isVazia = true;
        }
        else if(isPistaVazia && isVazia){
            sleep_for(0.25s);
               cout << endl << "TODOS OS AVIOES POUSARAM/DECOLARAM! Sem
trafego aereo no momento\n";
            sleep_for(0.25s);
        }
        else
        {
            sleep_for(0.25s);
                                cout << endl << "-------------PISTAS
OCUPADAS--------------\n" << endl;
            sleep_for(0.25s);
            for(i = 1; i <= quantAvioes; i++)
            {
                                imprimePistaDePouso(i, tipoNaPista[i-1],
aviaoNaPista[i-1]);
            }
            quantAvioes = 0;
        }
        sleep_for(1.25s);
    }
}

void imprimePistaDePouso(int quant, char tipo, int idAviao)
{
    if(tipo == 'D')
    {
           cout << "PISTA " << quant << " sendo usada pelo aviao " <<
idAviao << " >>>" << endl;
    }

    if(tipo == 'P')
```

```
    {
            cout << "PISTA " << quant << " sendo usada pelo aviao " <<
idAviao << " <<<" << endl;
    }
    sleep_for(0.5s);
}
```

```
#include <chrono>
#include <thread>
using namespace std::this_thread;
using namespace std::chrono_literals;

struct node {
    int info;
    int prior;
    char tipo;
    node *link;
};

node *inicializaFP(node *L)
{
    L = NULL;
    return L;
}

node *insereFP(node *L, int valor, char tipo, int prior)
{
    node *N, *P, *ANT;

    N = new node;
    N->info = valor;
    N->prior = prior;
    N->tipo = tipo;

    if (L == NULL) {
        L = N;
        N->link = NULL;
    }
    else {
        P = L;

        while ((P != NULL) && (prior >= P->prior)) {
```

```
            ANT = P;
            P = P->link;
        }
        if (P == L) {
            N->link = L;
            L = N;
        }
        else {
            ANT->link = N;
            N->link = P;
        }
    }
    return L;
}

node *removeFP(node *L, int *n, char *tipo, int *prior) {
    node *AUX;

    if (L != NULL) {
        *n = L->info;
        *prior = L->prior;
        *tipo = L->tipo;
        AUX = L;
        L = L->link;
        delete AUX;
    }
    return L;
}

int verificaSeVazia(node *L) {
    if (L == NULL)
        return 1;
    else
        return 0;
}

void exibe(node *L)
{
    node *P = L;
    while (P != NULL) {
         cout << "|AVIAO: " << P->info << " |PRIORIDADE: " << P->prior
<< " |REQUISITA: " << P->tipo << "|\n" ;
        P = P->link;
```

```
        sleep_for(0.025s);
    }
}

node* insereFilaChegada(node *L, int data, char tipo ,int prior) {
    node *P, *N;

    N = (node *) malloc (sizeof(node));
    N->info = data;
    N->prior = prior;
    N->tipo = tipo;
    N->link = NULL;

    if (L == NULL){
        L = N;
    }
    else {
        P = L;

        while(P->link != NULL) {
            P = P->link;
        }
        P->link = N;
    }
    return L;
}
```