Relatório do ep0 - mac323

Davi de Menezes Pereira, 11221988 16/03/2020

Como foi feito o Exercício Programa

Breve explicação de como eu entendi o enunciado do ep e como fiz o meu programa (sem muitos detalhes).

Primeiramente, para cada instante t de tempo, o programa lê um número K, o número de aviões entrando em contato naquele momento. Depois disso, ele envia cada avião para sua respectiva fila.

Foram feitas duas filas principais, uma para aviões de emergência e outra para aviões "normais" (que não são de emergência) e na classe dos aviões foi deixado um bool para dizer se o avião é de emergência ou não. Ou seja, não há especificação sobre qual é o tipo da emergência (isto vale para todo o programa: as emergências foram tratadas igualmente porque isso não estava muito claro no enunciado).

Depois disso, temos que mandar os aviões paras as pistas, que são representadas no código por três variáveis do tipo "Plane". Para fazer isso, eu checo quanto tempo o avião "está" na pista (considerando que cada avião gasta 3 tempos para fazer uma ação) e, além disso, checo o tempo estimado do avião. Esse útlimo passo foi uma alternativa encontrada para evitar que o avião seja enviado mais de uma vez pra fila dos que pousaram ou decolaram (para cada avião que já foi mandado pra lista de pouso ou decolagem, eu coloco um valor negativo no tempo estimado). Caso a pista esteja livre, eu coloco primeiro os aviões de emergência, e, se não tiver mais aviões de emergência, coloco algum da fila normal (nesse passo há pequenas alterações para o caso da pista 3). Para maior eficiência na alocação dos aviões, essa etapa é feita primeiramente para a pista 3.

Sobre a eficiência do programa: No enunciado do ep foi pedido pra fazer o gerenciamento das pista de forma eficiente, entretando devido a algumas restrições do enunciado e alguns detalhes não terem ficado muito claros, optei por fazer dessa forma explicada acima (com duas filas principais). Não acho que é a forma mais eficiente, porém se for considerado o número de aviões que devem chegar para 3 pistas, essa maior eficiência não iria ter um impacto tão grande no tempo gasto pelo programa.

Sobre a organização do programa: foram feitas classes para os aviões e uma estrutura de fila, embora essa estrutura não seja a tradicional, uma vez que foram necessárias algumas

funções auxiliares que não existem numa fila tradicional. Ainda sobre as filas, optei por não usar template, considerando que a fila só foi usada para um tipo conhecido. Sobre a classe dos aviões, optei por deixar os atributos como público.

Formato do input

Listing 1: Parte do código em plane.h.

```
1     class Plane
2     {
3         public:
4             string id;
5             string from_to;
6
7             bool emergency;
8
9             int estimated_time;
10            int waiting_time;
11            /*auxiliar variable to control the time when the flight track ↩
                  will be ocupied*/
12            /*on this atribute, everyone begins with 3*/
13
14        public:
15            Plane(); /* default constructor */
16            Plane(string t_id, string t_from_to, bool t_landing, bool ↩
                  emergency, int t_fuel, int t_estimated_time);
17            /*~plane();*/
18            int track_time;
19            int fuel;
20            bool landing;
21
22            bool isLanding(Plane p);
23    };
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Listing 2: Parte do código em airport.cpp que lê os aviões (C++).

```
cin >> K;
for(k = 0; k < K; k++)
{
    cin >> t_id;
    cin >> t_from_to;

    cin >> t_landing;
    cin >> t_emergency;

    cin >> t_fuel;
    cin >> t_estimated_time;

    Plane aux {t_id, t_from_to, t_landing, t_emergency, t_fuel, ↩
       t_estimated_time};

    if(t_emergency) emergencies.push(aux);
    else normal.push(aux);
}
```