ATRASOS NAS FILAS DE SEGURANÇA EM AEROPORTOS

Algoritmia e Programação

2023/24

O atraso e a aglomeração nas filas de segurança nos aeroportos impactam negativamente a experiência dos passageiros e a pontualidade dos voos.

Aluna: Kássia Eduarda Guimarães
Nº de aluna: A105755

Braga, November 18, 2023

Índice

Kássia Eduarda Guimarães
nª: A105755

Introdução

É notável que inúmeras pessoas enfrentam atrasos e, por vezes, perdem seus voos devido às extensas filas na área de segurança dos aeroportos e à aglomeração na sala de espera antes do embarque. Grande parte desse conflito pode ser atribuída à falta de informações precisas e ao conhecimento sobre os horários ideais para o processo de segurança. Isso frequentemente resulta na aglomeração de passageiros nas filas e nas áreas de espera, impactando negativamente a experiência de viagem,

Quando fala-se sobre as 'filas de segurança' nos aeroportos, está sendo referido as filas específicas onde todos os passageiros devem passar antes de embarcar em seus voos. Essas filas são projetadas para garantir a segurança de todos a bordo e incluem a passagem pelo controle de segurança, onde seus pertences são examinados por raio-X e você passa por procedimentos de triagem.

Nesse contexto, o projeto é destinado a otimizar o fluxo de passageiros durante o processo de segurança, incluindo a gestão eficaz das filas, e a melhorar a comunicação das informações relevantes pode ter um impacto significativo na experiência de viagem e na pontualidade dos voos. Contendo duas interfaces, o menu contempla a parte do passageiro, em que deve inserir seus dados para conseguir obter os horários para passar pela segurança, e a parte do funcionário, onde são alocados os dados dos voos de saída durante o dia no aeroporto.

Parâmetros de entrada dos passageiros

1. Dados pessoais;
    1.1. Nome.
    1.2. Número de passaporte.
2. Número do voo.
3. Horário do voo.

Parâmetros de entrada dos funcionários

1. Senha fictícia de acesso ao sistema.
2. Número de aeronaves do dia.
3. Informações dos voos do dia, de acordo com o número de aeronaves.

Com base nestes parâmetros de entrada, a aplicação a desenvolver deve ter as seguintes funcionalidades:

Funcionalidades

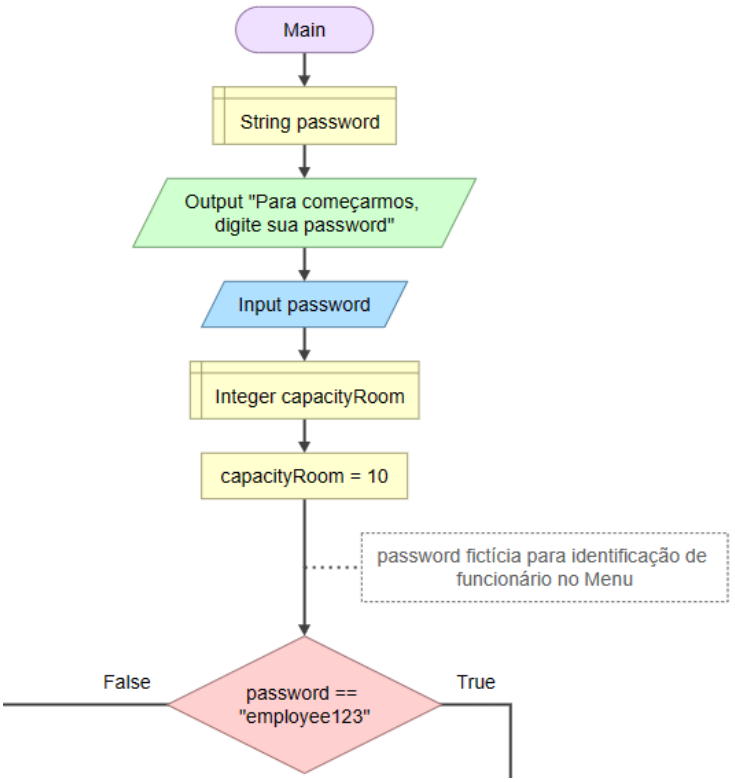
- **Funcionalidade 1:** *Registro de Passageiros*; coleta de informações dos passageiros, incluindo nome, número de identificação e número do voo.
- **Funcionalidade 2:** *Capacidade do Salão de Espera;* determinação da capacidade máxima do salão de espera após a passagem pela segurança.
- **Funcionalidade 3:** *Horário de Segurança;* estabelecimento dos horários ideais de chegada à fila de segurança e espera antes do voo.
- **Funcionalidade 4:** *Aeronaves no dia;* coleta de dados dos voos que saem do aeroporto no dia.

Kássia Eduarda Guimarães
nª: A105755

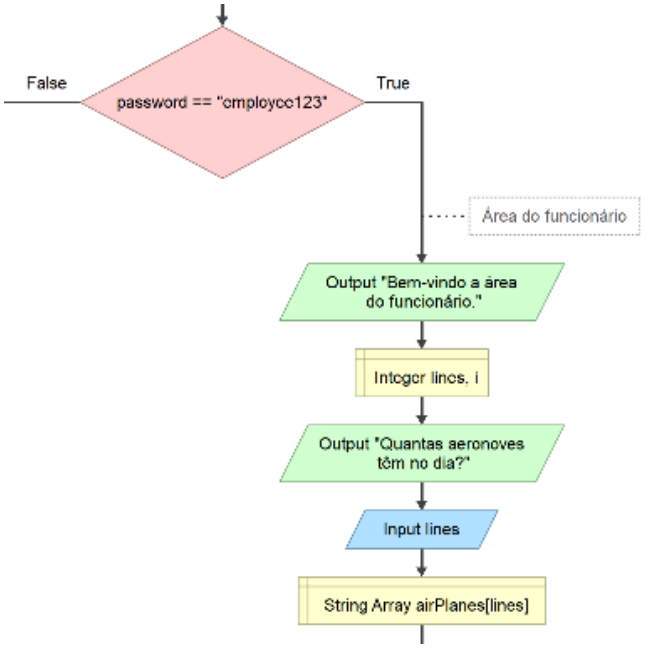
Variáveis e seus tipos

Main

Na primeira parte "Main" é apresentado um login para autenticar quem são os funcionários e os passageiros para, então, realizarem seus inputs.

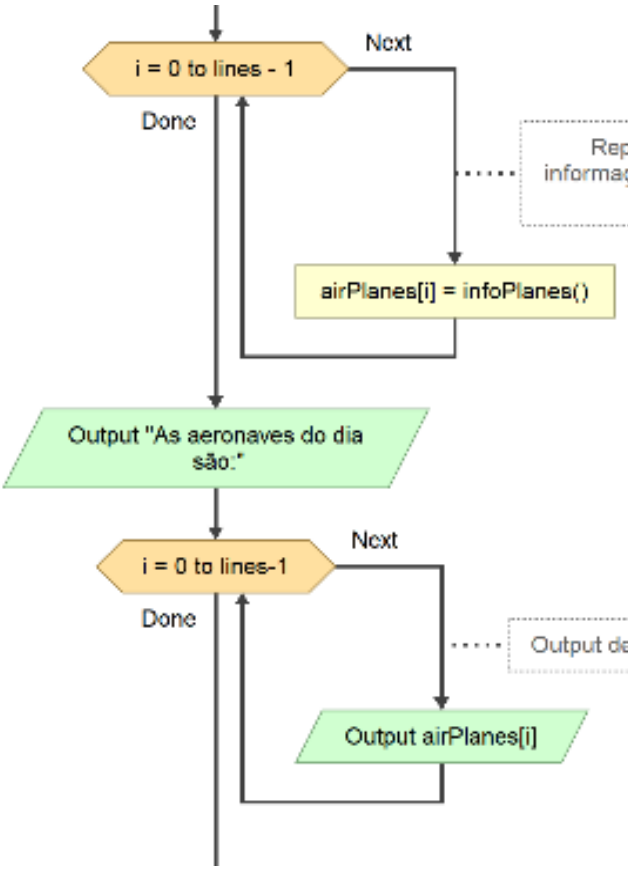


Começa sendo declarada uma variável *"password" do tipo string* que indica uma senha para identificação do funcionário, sendo imposta como "employee123". Logo após o input da password é feita uma condição para a verificação. Se a password dada for igual a "employee123", então o usuário é direcionado para a "*Área do Funcionário*", senão é direcionado para a "Área do Passageiro". A segunda variável declarada é *"capacityRoom" do tipo integer* que indica a capacidade do salão de espera para o voo, sendo imposta com o valor 10 para efeitos de teste.



Área do Funcionário

A "*Área do Funcionário*" é iniciada dando um output indicando que o usuário está na área do funcionário. Logo após é declarada uma variável *"lines" do tipo integer* que indica o número de linhas de um array posteriormente declarado, a variável indica o número de aeronaves que o aeroporto terá durante todo o dia e, junto a esta, é declarado *"i" do tipo integer* que funciona como contador para o for, posteriormente introduzido. Depois é declarado um *array "airPlanes" do tipo string* onde vão ser alocados às



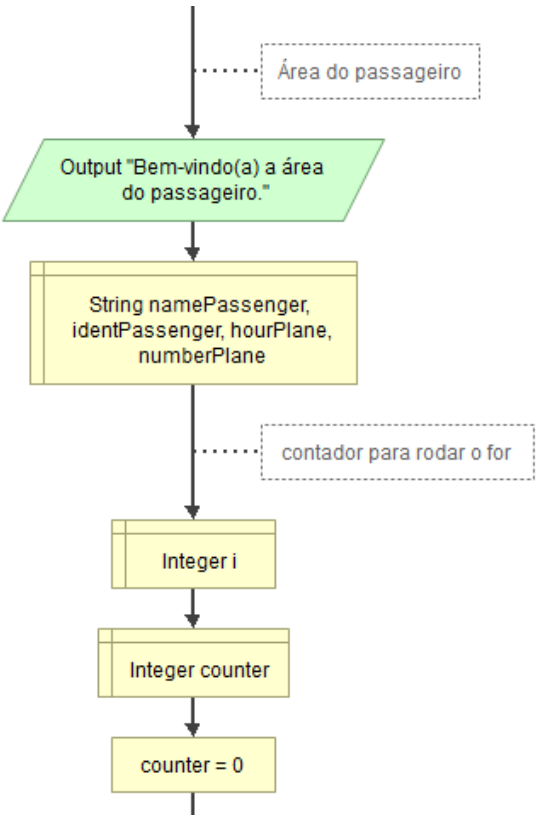
aeronaves que passam pelo aeroporto durante o dia.

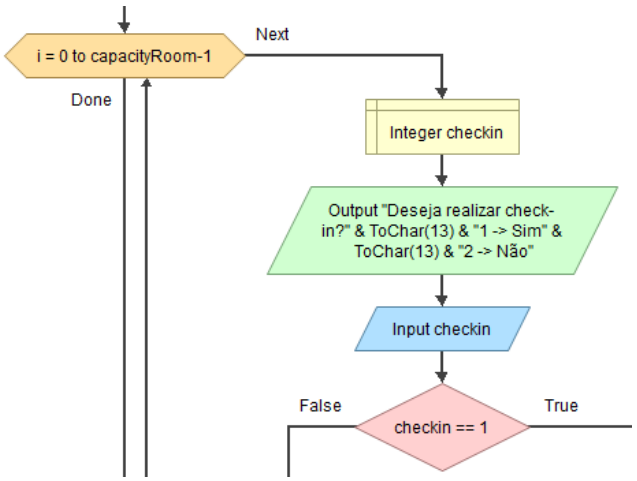
Logo após é dada uma estrutura de *repetição for, com o contador i iniciado em 0 e indo até (lines-1)*, enquanto o i não for igual à (lines-1) vai alocando valor no array airPlanes[i], com o índice i (sendo iniciado em 0), valor dado na *função "infoPlanes"*.

Quando a estrutura de repetição for i in range(0, (lines-1)) for finalizada é dado um output com todas as aeronaves do dia, sendo feitos prints através da estrutura de *repetição for, com o contador i iniciado em 0 e indo até (lines-1)*.

Área do Passageiro

A "*Área do Passageiro*" é iniciada dando um output indicando que o usuário está na área do passageiro. Logo após são declaradas as variáveis *"namePassenger", indicando o nome do passageiro, "identPassenger", indicando o número do passaporte para identificação, "hourPlane", para indicar o horário do voo, e "numberPlane", contemplando o número do voo, todas do tipo string*, sendo alocado valor à todas posteriormente. Depois são declaradas as variáveis *"i" sendo o contador para a estrutura de repetição, e "counter", servindo como contador de pessoas que realizaram check-in para entrar na sala de espera pelo voo sendo indicado valor inicial como 0, ambas do tipo integer*.





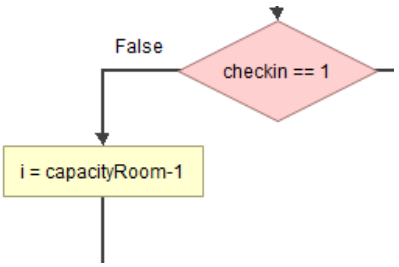
Seguindo é dada uma estrutura de *repetição for, com o contador i iniciado em 0 e indo até (capacityRoom-1)*, no início é declarado uma variável *"checkin" do tipo integer*, que vai servir como input para a realização do check-in do passageiro, logo após é dado um output perguntando se a pessoa deseja realizar o check-in ou não, sendo 1 para "Sim" e 2 para "Não". Com a decisão do usuário é feita uma condicional.

Se a pessoa desejar realizar o check-in na fila da segurança. Os primeiros passos para a realização do check-in são apanhar o nome, número de identificação, hora do voo e número do voo de cada passageiro, todos retornados como funções.

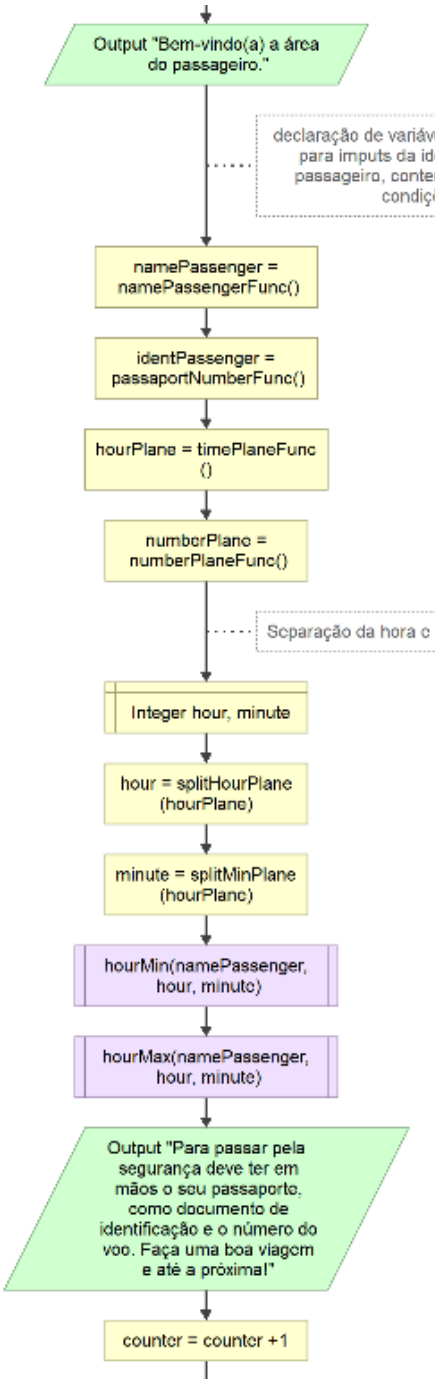
Depois de receber os dados obrigatórios para o check-in é feita a separação da parte da hora e minuto para o cálculo da hora máxima e hora mínima de entrada na fila, sendo declaradas as variáveis *"hour" e "minutes" do tipo inteiro*, retomadas com o resultado das funções "splitHourPlane" e "splitMinPlane", respectivamente.

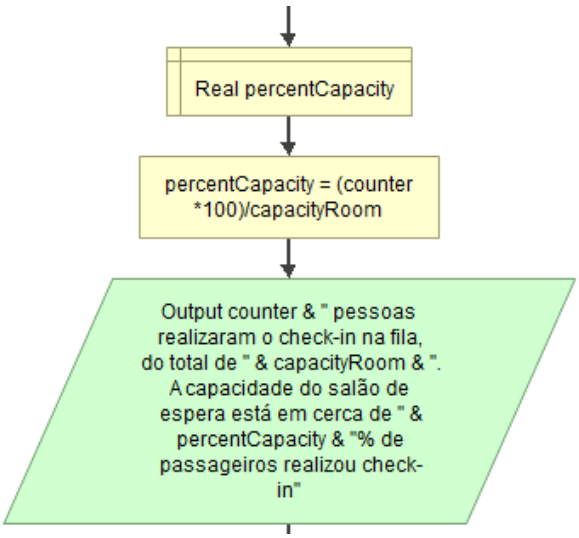
Para, então, dar de resposta ao usuário a mora mínima e a hora máxima para chegar na fila é feito um chamamento das funções "hourMin" e "hourMax".

Considerando que é necessário saber o número de pessoas que realizaram o check-in na fila, foi posto um counter para determinada situação, onde, para cada ciclo for, em que o passageiro realiza o check-in, é contabilizado.



Relembrando a condição para a estrutura de repetição, o ciclo é repetido até chegar ao número máximo de pessoas na sala, declarado como "capacityRoom" ou, conforme a condicional, não forem realizados mais check-ins.





Quando o ciclo "for" é finalizado é dado um retorno com o número de pessoas que realizaram o check-in e a porcentagem da capacidade do salão de espera pelo voo.

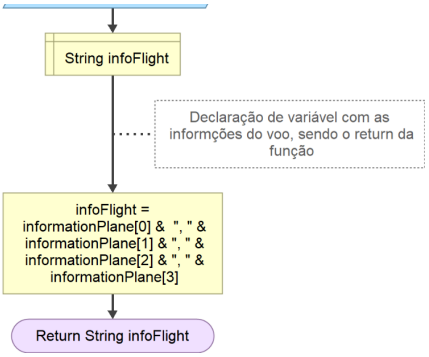
Inicia-se declarando a variável *"percentCapacity" do tipo real*, que é retomada com o cálculo $(counter * 100)\backslash capacityRoom$, sendo o contador de pessoas, multiplicado por 100 e depois dividido pela capacidade total do salão, retomando a capacidade atual do hall e finalizando o Main.

Funções

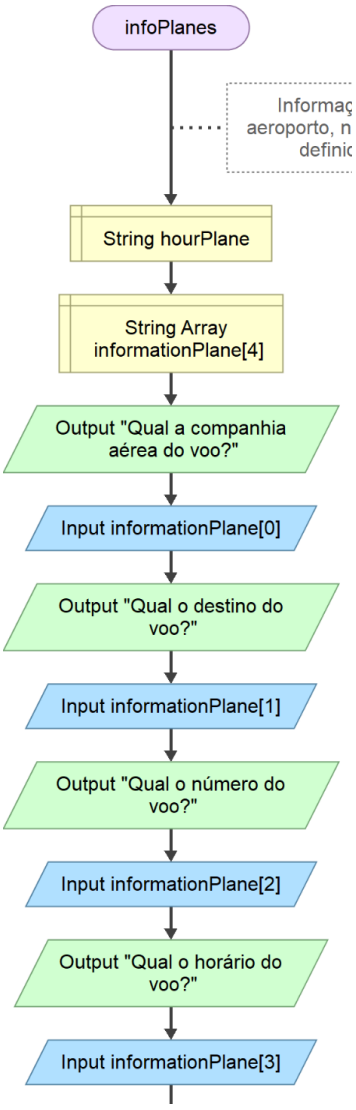
Informações dos voos

A função *infoPlanes()* tem como intuito capturar as informações dos voos diários, descrito pelo funcionário, e colocar no array *airPlane declarado no Main*.

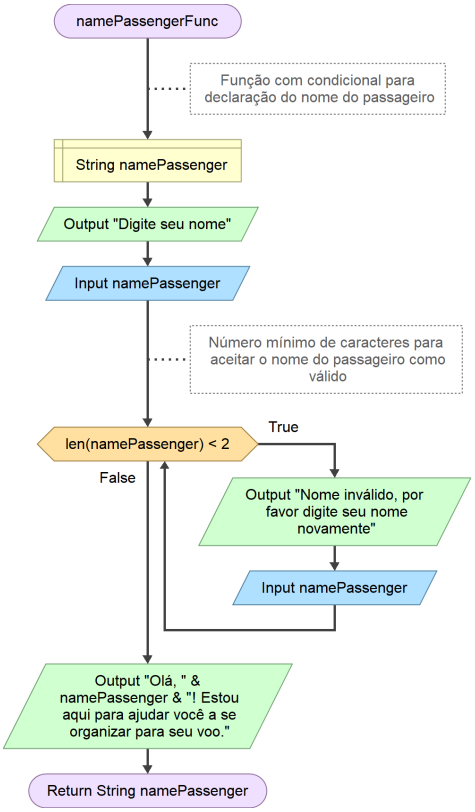
Começa por ser declarada a variável *"hourPlane" do tipo string* e depois um array *"informationPlane" do tipo string com 4 linhas*, número de linhas pré-definido com base nos parâmetros de: companhia aérea, alocado no índice 0, destino do voo, no índice 1, o número do voo, no índice 2, e o horário do voo, no índice 3.



Depois de dados os inputs necessários para a informação, o array é declarado em uma variável *"infoFlight" do tipo string*, para poder servir de retorno para a função e, então, ser alocada em uma linha do array *"airPlanes" do tipo string* no Main.



Identificação do passageiro



A função *namePassenger()* serve para pegar o nome do passageiro, recebe como input *"namePassenger" do tipo string*, que recebe uma condição para verificação, na forma de estrutura de repetição while, com a condição de aprovação quando "namePassenger" tiver tamanho maior ou igual a 2, tal condição foi imposta para reduzir a margem de erro na escrita no nome, como o nome de uma pessoa não pode conter menos que duas letras.

De forma semelhante a função *passportNumberFunc()* serve para pegar o número de identificação do passaporte, recebe como input *"passaportNumber" do tipo string*, condição para verificação e aprovação quando "passaportNumber" tiver tamanho igual a 8, já que a identificação do passaporte é dado com 8 valores, sendo 2 letras e 6 números.

A função *timePlaneFunc()* serve para pegar o número do voo, recebe como input *"timePlane" do tipo string*, condição para verificação e aprovação quando "timePlane" tiver tamanho igual a 5, já que a hora deve conter HH:MM, totalizando 5 caracteres.

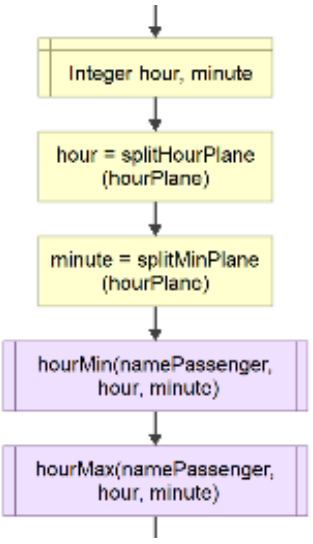
A função *numberPlaneFunc()* serve para pegar o número do voo, recebe como input *"numberPlane" do tipo string*, condição para verificação e aprovação quando "numberPlane" tiver tamanho igual a 5, imposta para deixar como obrigatoriedade cada número de voo conter apenas 5 caracteres.

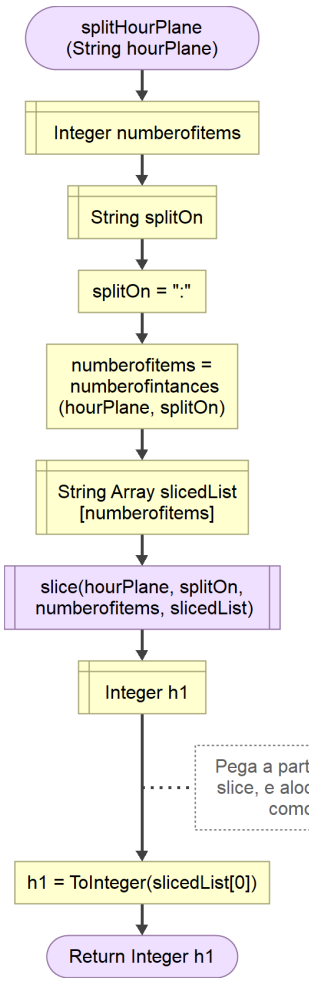
Split

As funções "split" foram criadas para fazer a separação da variável *"hourPlane" do tipo string*, entre a parte da hora e parte do minuto.

A função *splitHourPlane(hourPlane)* tem o intuito de separar a parte das horas, recebendo como parâmetro a string "hourPlane" (declarada anteriormente no main). De forma idêntica também é imposta a função *splitMinPlane(hourPlane)* tem o intuito de separar a parte dos minutos da mesma string.

Cada uma delas atua de forma análoga em sua composição, trago como exemplo a função *splitHourPlane(hourPlane)* para o melhor entendimento.





A função começa declarando a variável *"numberofitems" do tipo integer*, que vai servir para a contagem de itens, recebendo como valor o retorno da função "numberofintances", logo após é declarada a variável *"splitOn" do tipo string*, que vai já ser alocado o seu valor como ":", já que os dois pontos (:) é o critério de separação da parte da hora e parte dos minutos.

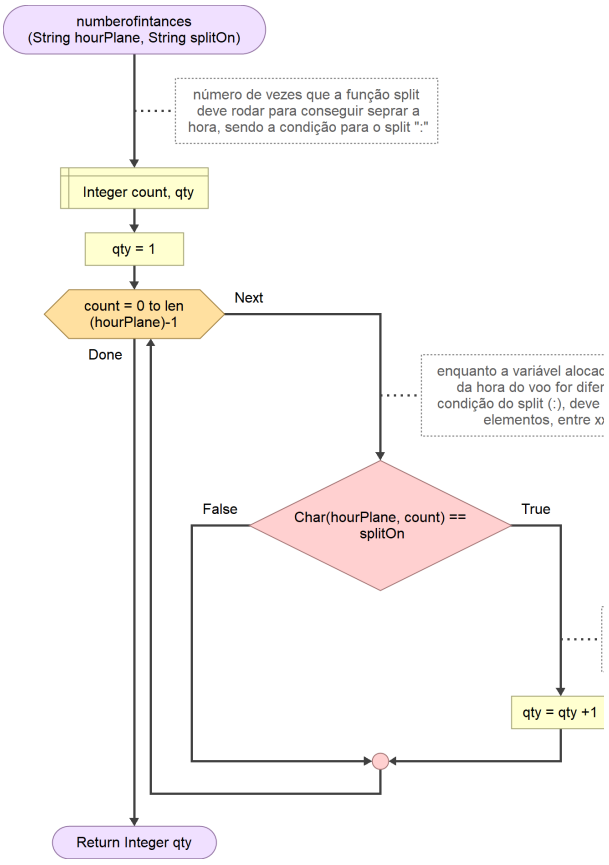
Sendo declarado, então, um array *"slicedList" do tipo string com tamanho igual ao número de itens a serem separados*, que vai servir para alocar os elementos de separação da minha string "hourPlane". Depois é feito uma chamada da função *"slice", com os parâmetros hourPlane, splitOn, numberofitems e slicedList*, onde é feita a real separação das variáveis.

Para alocar o valor da hora e ter como retorno da função, é feita a declaração da variável *"h1" do tipo integer*, que remete a parte da hora da string "hourPlane", já convertida para integer.

De forma semelhante, na função *splitMinPlane(hourPlane)* a variável de retorno se chama *"min1" do tipo integer.*

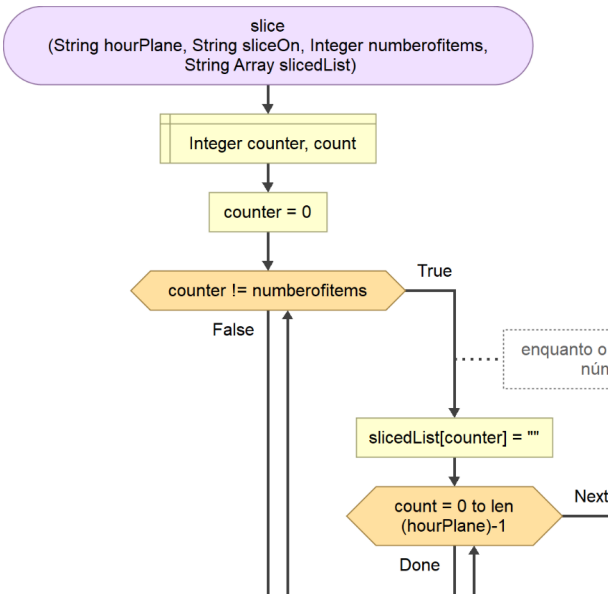
Número de instâncias

A função *numberofintances(hourPlane, splitOn)* tem como intuito dar como retorno a posição exata em que o caracter ":" está localizado. Começa sendo declarado um *"count"*, servindo como contador para o ciclo for, *e "qty" do tipo integer*, iniciando em 1 e servindo dar a quantidade de elementos a serem separados.



A estrutura de *repetição for, com o contador counter iniciado em 0 e indo até 4 (len(hourPlane)-1)*, depois é dada uma condicional que é dado for feita a verificação se caracter na posição 0 é igual ao "splitOn" (:).

Já que o formato da hora é dado como HH:MM, então ele vai rodar na posição 0 da string HH:MM, sendo, então, o caracter H e diferente de ":", roda novamente. Quando chegar no número de itens separados da string, vai, finalmente, ser igual à ":", alocando o valor $qty = qty + 1$ que é o valor 2.

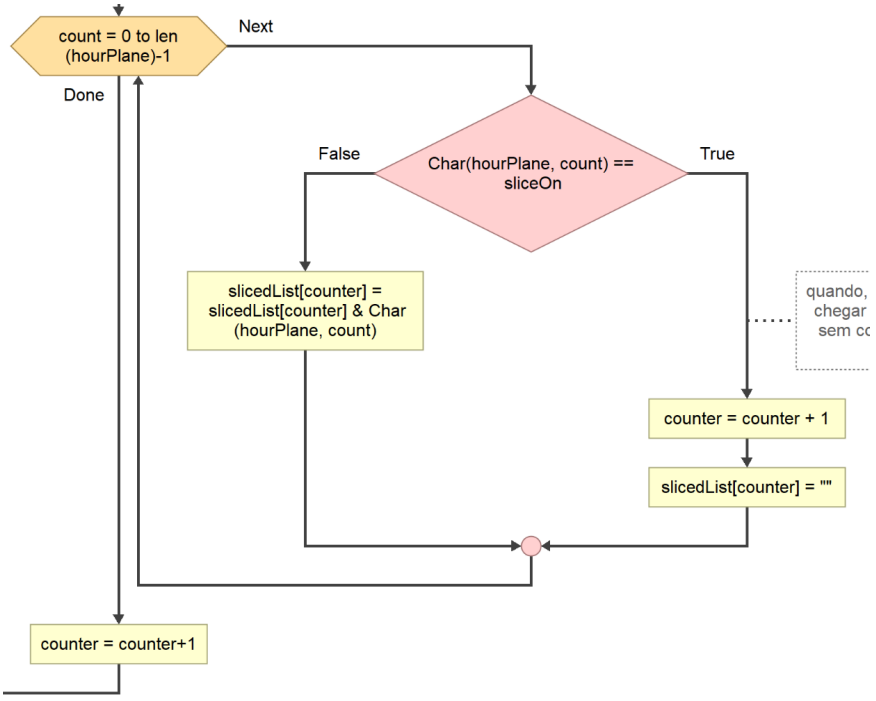


Separação das instâncias

A função *"slice(hourPlane, splitOn, numberofitems, slicedList)"* tem como intuito separar as instâncias da hora e minutos e alocar num array "slicedList".

Começa sendo declarado um *"count"*, servindo como contador para o ciclo while, *e "count" do tipo integer*, servindo como contador para o segundo ciclo for.

A estrutura de *repetição while, que vai até o contador "counter" ser igual ao "numberofitems" que é 2* , depois é levado para uma estrutura de *repetição for, que vai até o contador "count" iniciado em 0 e indo até 4 (len(hourPlane)-1)*.



Depois é dada uma condicional que é dado for feita a verificação se caracter na posição 0 é igual ao "splitOn" (:).

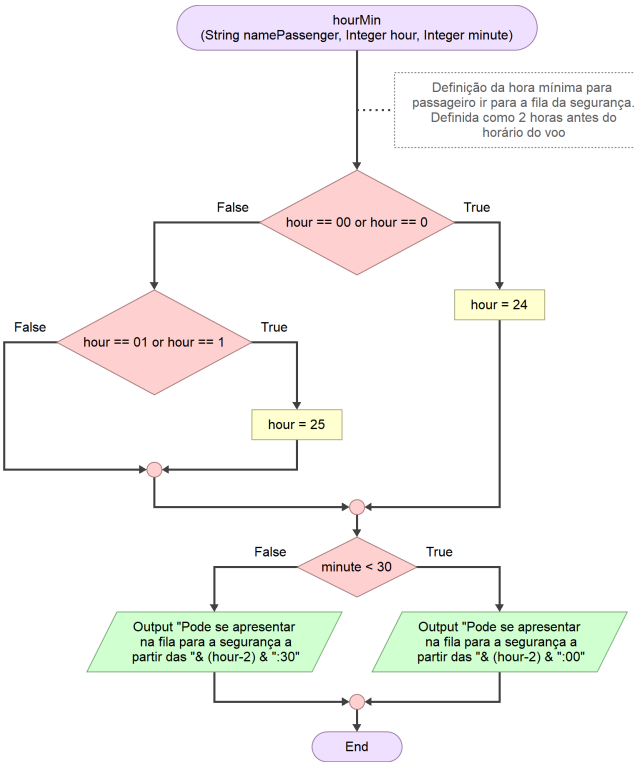
Já que o formato da hora é dado como HH:MM, então ele vai rodar na posição 0 da string HH:MM, sendo, então, o caracter H e diferente de ":", alocando no array "slicedList[0]" o caracter na posição 0, rodando novamente. Quando chegar no número de itens separados da string for, finalmente, igual à ":", alocando o valor $counter = counter + 1$, sendo alocado valor "", vazio. Rodando novamente até terminar de percorrer toda a string. Dando como retorno $slicedList[0] = HH$ e $slicedList[1] = MM$.

Hora mínima

Tem como finalidade declarar a hora máxima que a pessoa pode ir para a fila da segurança, sendo imposta como hora mínima 2 horas antes do horário do voo, com suas margens de erro. A função *hourMin(namePassenger, hour, minute)* começa a sendo imposta uma condição se a hora for igual a 00, o valor da hora vai ser 24, senão, se a hora for igual a 01, o valor da hora vai passar a ser 25, ou continua tudo igual, sem modificações. Essa primeira condição foi imposta porque se a hora do voo for 00, por exemplo, quando subtrair

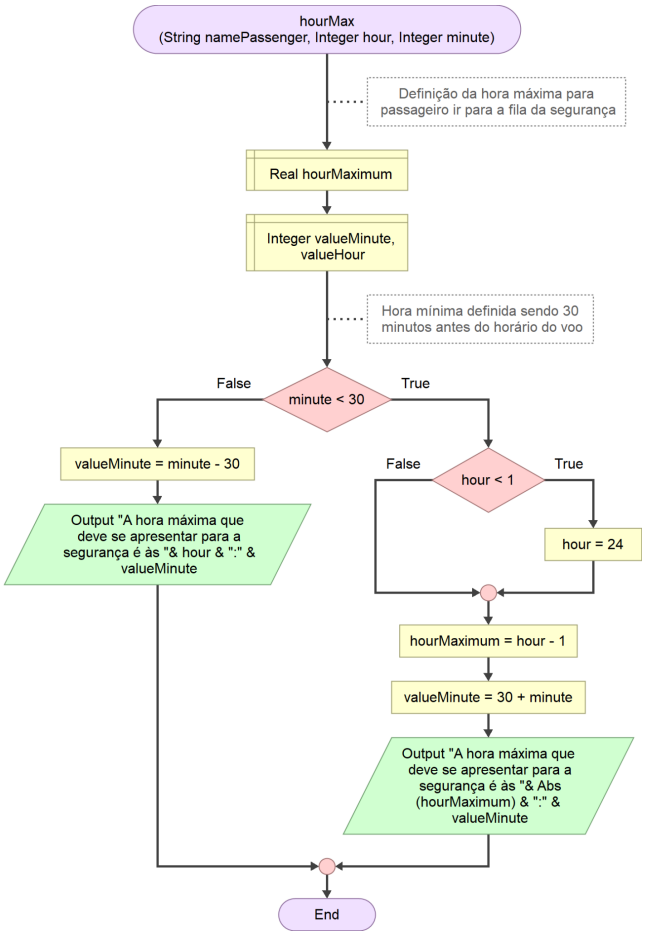
2 valores (2 horas), vai ser -2, mas o formato hora não tem como ser -2, mas, se for alocado o valor de $00 \rightarrow 24$ para dizer a hora mínima vamos ter o valor de 22, o mesmo serve para a hora sendo $01 \rightarrow 25$, sendo a hora mínima 23.

Depois de feita essa verificação da parte da hora, é feita a verificação da parte do minuto. Se a hora do voo for menor do que 30, então a pessoa pode chegar na HH:00, dando uma margem maior para o tempo de entrada, se for maior do que 30 então a pessoa pode chegar na HH:30, com o mesmo intuito.



Hora máxima

Tem como objetivo declarar a hora máxima que a pessoa pode ir para a fila da segurança, sendo imposta como 30 minutos antes do horário do voo. A função *hourMax(namePassenger, hour, minute)* começa sendo declaradas as variáveis *"hourMaximum"*, para alocar o valor da hora máxima quando o minuto for menor do que 30, *e "valueMinute" do tipo integer,* que calcula a parte dos minutos.



Para ser feito o cálculo, primeiro é imposta uma condicional, se a parte do minuto for menor do que 30, então passa para outra condicional, se a hora for menor do que zero colocasse $hour = 24$, para conseguir subtrair 1 hora, senão continua normalmente. Quando a parte do minuto for menor do que 30 é calculada a hora máxima, "hourMaximum" sendo $hour - 1$ e os minutos calculados como "valueMinute" sendo $minute + 30$

Caso o minuto do horário for maior do que 30, então, simplesmente, é calculado o valor $minute - 30$.

Código em Python

Para realizar o código em python foram feitas algumas alterações no código exportado do flowgorithm, para melhor execução do projeto. Foram retiradas 4 funções que realização o “split” em python, as funções “numberofintances”, “splitHourPlane”, “splitMinPlane” e “slice” foram excluídas. Segue a demonstração do código da parte modificada, decorrendo como a maior e principal mudança.

```
if checkin == 1:

# declaração de variáveis com funções para inputs da identificação
do passageiro, contendo condições condições

            namePassenger = namePassengerFunc()

            identPassenger = passaportNumberFunc()

            hourPlane = timePlaneFunc()

            numberPlane = numberPlaneFunc()

            # Separação da hora e minuto

            splitTime = hourPlane.split(":")

            hour = int(splitTime[0])

            minute = int(splitTime[1])



            hourMin(namePassenger, hour, minute)

            hourMax(namePassenger, hour, minute)
```

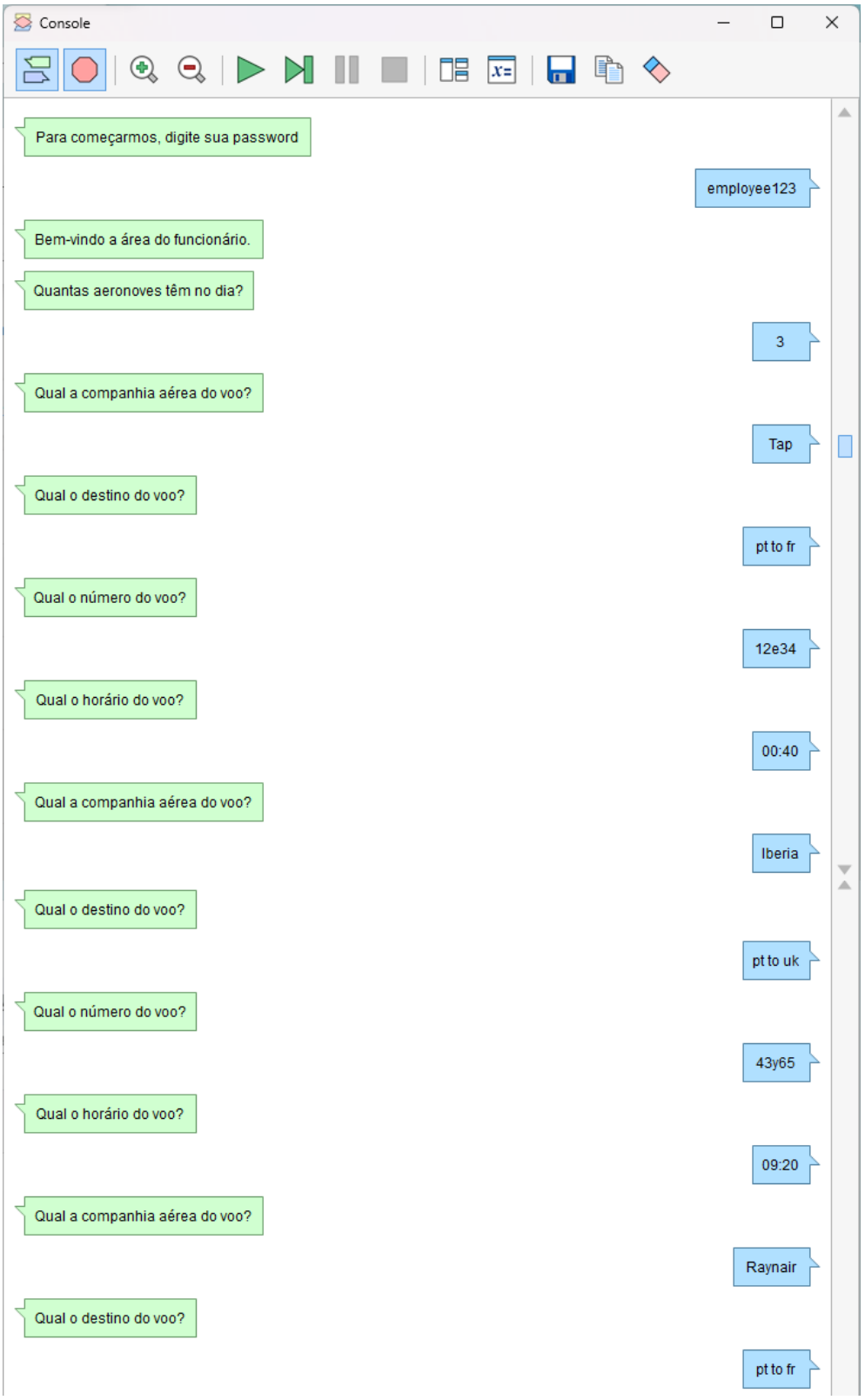
Depois foram feitas umas modificações nos “prints”. No flowgorithm para haver quebra de linha era preciso fazer o “ToChar(13)” onde em python só é necessário o “\n”. Segue demonstração do código.

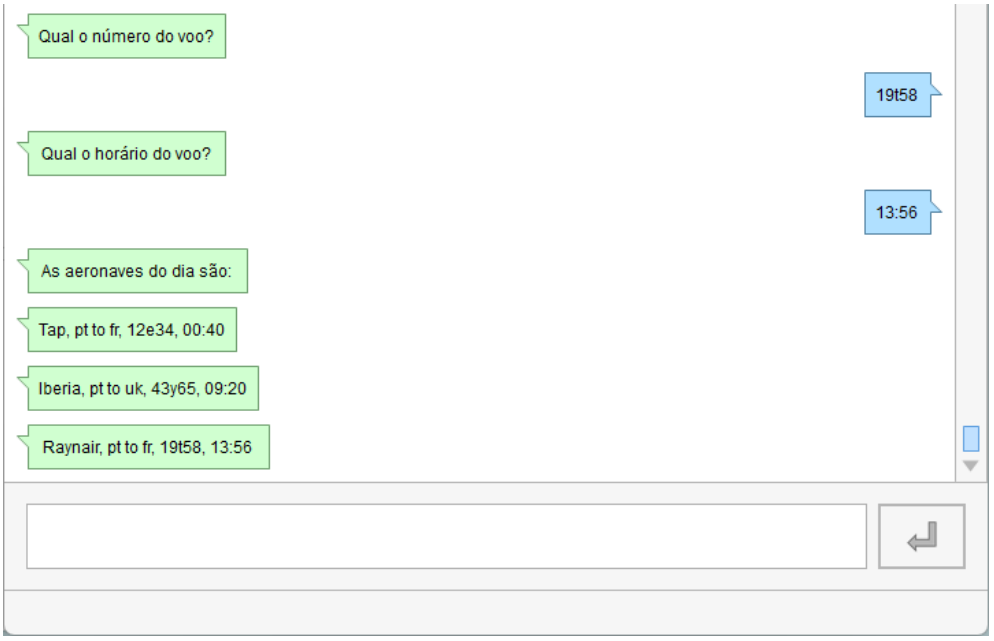
```
        print("Deseja realizar check-in?\n1 -> Sim\n2 -> Não")

        checkin = int(input())
```

Demonstração de testes

Área do Funcionário




Qual o número do voo?
19t58
Qual o horário do voo?
13:56
As aeronaves do dia são:
Tap, pt to fr, 12e34, 00:40
Iberia, pt to uk, 43y65, 09:20
Raynair, pt to fr, 19t58, 13:56

Kássia Eduarda Guimarães
nª: A105755

Área do Passageiro