BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

4° PERÍODO 2021/01

DISCENTES:

BERNARDO RESENDE ANDRÉS

CLAUDIMAR JOSÉ DA CRUZ

RAFAEL DE SOUZA DAMASCENO

ESTRUTURAS DE DADOS

Atividade da 1° etapa para a aprovação da disciplina de Estrutura de dados, ministrada por Prof. Nairon Neri Silva.

UNIVERSIDADE PRESIDENTE ANTÔNIO CARLOS

Barbacena – 2021

Descrição da Atividade

A atividade proposta tem como objetivo a criação de um algoritmo que simula o funcionamento de um aeroporto, para atender decolagens e aterrissagens (incluindo casos de emergência). Para tal usaremos o conteúdo visto na matéria de Fila Encadeada.

Pede-se na atividade a utilização de duas filas, representando os aviões que irão decolar e os que irão pousar, sempre que houver uma emergência ela terá prioridade para o pouso. É necessário utilizar três pistas, que poderão alterar entre pouso e decolagem, cada avião será representado por um id, onde par significa decolagem e ímpar de aterrissagem. Tanto a escolha das pistas que serão usadas, quanto a quantidade de aviões criados, as suas emergências serão controladas randomicamente. As pistas terão que ter um controle para não crescerem excessivamente, e os aviões que solicitarem emergência irão para o início da fila e receberão um asterisco para identificá-lo.

O programa será controlado pelo usuário através da tecla enter, a cada vez apertado será exibido para ele os conteúdos das filas, as pistas e o que cada uma está recebendo, total de aviões atendidos e total de emergências.

Bibliotecas utilizadas

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Além das tradicionais bibliotecas *<stdio.h>* e *<stlib.h>,* utilizamos também a *<time.h>*, para conseguirmos utilizar números randômicos (srand(time(NULL))).

Funções utilizadas

```c
typedef struct item
{
    int id;
    int emergencia;
    struct item *prox;

}node;
```

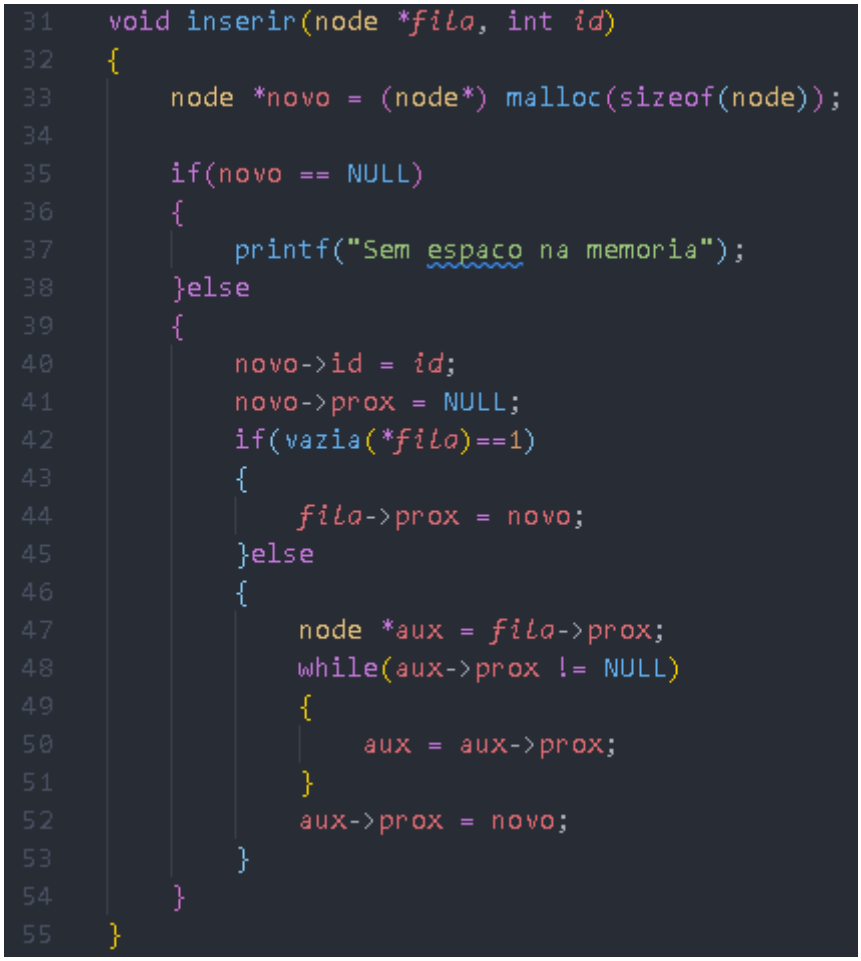
Criamos o typedf struct para armazenar os id's dos aviões, suas respectivas emergências e o ponteiro para o próximo elemento da fila.

```c
void criarFila(node *fila)
{
    fila->prox = NULL;
}
```

Função responsável por gerar a fila .

```c
int vazia(node fila)
{
    if(fila.prox == NULL)
    {
        return 1;
    }else
    {
        return 0;
    }
}
```

Função responsável por verificar a fila, se o retorno for igual a 1, significa que a fila está vazia, .caso o retorno seja 0, ela não está vazia.



```c
void inserir(node *fila, int id)
{
    node *novo = (node*) malloc(sizeof(node));

    if(novo == NULL)
    {
        printf("Sem espaco na memoria");
    }else
    {
        novo->id = id;
        novo->prox = NULL;
        if(vazia(*fila)==1)
        {
            fila->prox = novo;
        }else
        {
            node *aux = fila->prox;
            while(aux->prox != NULL)
            {
                aux = aux->prox;
            }
            aux->prox = novo;
        }
    }
}
```

Função responsável por inserir os id’s nas respectivas filas.

```
90  void imprimir(node fila_DEC, node fila_AT)
91  {
92      if(vazia(fila_DEC)==1)
93      {
94          printf("\nA fila de decolagem esta vazia!!!\n");
95      }else
96      {
97          node *aux = fila_DEC.prox;
98          printf("\n============================\n");
99          printf("FILA DE DECOLAGEM:  ");
100
101         while(aux != NULL)
102         {
103             printf("<- [%d] ", aux->id);
104             aux = aux->prox;
105         }
106     }
107     printf("\n============================\n");
108     printf("\n");
109
110     if(vazia(fila_AT)==1)
111     {
112         printf("\nA fila aterrissagem esta vazia!!!\n");
113     }else
114     {
115         node *aux = fila_AT.prox;
116         printf("\n============================\n");
117         printf("FILA DE ATERRISAGEM:  ");
118
119         while(aux != NULL)
120         {
121             if(aux->emergencia == 1)
122             {
123                 printf("<- [*%d]", aux->id, aux->emergencia);
124             }else
125             {
126                 printf("<- [%d]", aux->id, aux->emergencia);
127             }
128             aux = aux->prox;
129         }
130     }
131      printf("\n============================\n");
132      printf("\n");
133 }
```

Função responsável por imprimir a fila de decolagem, a fila de aterrissagem. Na linha 121, criamos um IF para inserirmos o (*) nos aviões que declararam emergência.

```c
int contador(node fila)
{
    int i =0;
    node *aux = fila.prox;

    while(aux != NULL)
    {
        i++;
        aux = aux->prox;
    }
    return i;
}
```

Função responsável por contar os elementos das filas.

```c
void PousoEmergencia(node *fila_AT, int id, int emergencia)
{
    node *novo = (node*) malloc(sizeof(node));

    if(novo == NULL)
    {
        printf("Sem espaco na memoria");
    }else
    {
        novo->id = id;
        novo->emergencia = emergencia;
        novo->prox = NULL;
        if(vazia(*fila_AT)==1)
        {
            fila_AT->prox = novo;
        }else
        {
            node *atual = fila_AT->prox;
            node *ant = fila_AT;
            while((atual!= NULL) && (atual->emergencia <= novo->emergencia))
            {
                ant =atual;
                atual = atual->prox;
            }
            novo->prox = ant->prox;
            ant->prox = novo;
        }
    }
}
```

Função responsável por inserir os aviões que declararam emergência na primeira posição da fila de aterrissagem.

```c
void controlePista(node *fila_AT, node *fila_DEC)
{
    int pista =0;
    int pista2 =0;
    int pista3 =0;

    int i =0;
    int j =0;
    int k =0;
    int l =0;
    int m =0;

    int cont =0;

    for(i=0;i<1;i++)
    {
        if(vazia(*fila_AT) == 0)
        {
            node *aux1 = fila_AT->prox;
            fila_AT->prox = aux1->prox;
            pista = aux1->id;
            j =0;

            guardar_contadr = guardar_contadr + 1;

        }else if(vazia(*fila_DEC) ==0)
        {
            node *aux2 = fila_DEC->prox;
            fila_DEC->prox = aux2->prox;
            pista = aux2->id;
            j =1;

            guardar_contadr = guardar_contadr + 1;
        }
        if(vazia(*fila_AT) == 0)
        {
            node *aux3 = fila_AT->prox;
            fila_AT->prox = aux3->prox;
            pista2 = aux3->id;
            k =0;

            guardar_contadr = guardar_contadr + 1;

        }else if(vazia(*fila_DEC) ==0)
        {
            node *aux4 = fila_DEC->prox;
            fila_DEC->prox = aux4->prox;
            pista2 = aux4->id;
            k =1;

            guardar_contadr = guardar_contadr + 1;
        }
        if(vazia(*fila_DEC) == 0)
        {
            node *aux5 = fila_DEC->prox;
            fila_DEC->prox = aux5->prox;
            pista3 = aux5->id;
            l =0;

            guardar_contadr = guardar_contadr + 1;
```

```
213             }else if(vazia(*fila_AT) ==0)
214             {
215                 node *aux6 = fila_AT->prox;
216                 fila_AT->prox = aux6->prox;
217                 pista3 = aux6->id;
218                 l = 1;
219 
220                 guardar_contadr = guardar_contadr + 1;
221             }
222         }
223             imprimir(*fila_AT, *fila_DEC);
224 
225             if(pista == 0)
226             {
227                 printf("\nPista 1 - completamente vazia\n");
228 
229             }else if(j ==0)
230             {
231                 printf("\nPista 1 [%d] - aterrisando\n", pista );
232 
233             }else if(j ==1)
234             {
235                 printf("\nPista 1 [%d] - decolando\n", pista);
236             }
237             if(pista2 == 0)
238             {
239                 printf("\nPista 2 -completamente vazia\n");
240 
241             }else if(k ==0)
242             {
243                 printf("\nPista 2 [%d] - aterrisando\n", pista2 );
244 
245             }else if((k == 1) != 0)
246             {
247                 printf("\nPista 2 [%d] - decolando\n", pista2);
248             }
249             if(pista3 == 0)
250             {
251                 printf("\nPista 3 -completamente vazia\n");
252 
253             }else if(l ==0)
254             {
255                 printf("\nPista 3 [%d] - decolando\n", pista3 );
256 
257             }else if(l ==1)
258             {
259                 printf("\nPista 3 [%d] - aterrisando\n", pista3);
260 
261             }
262             printf("\nAvioes atendidos: %d\n", guardar_contadr);
263     }
```

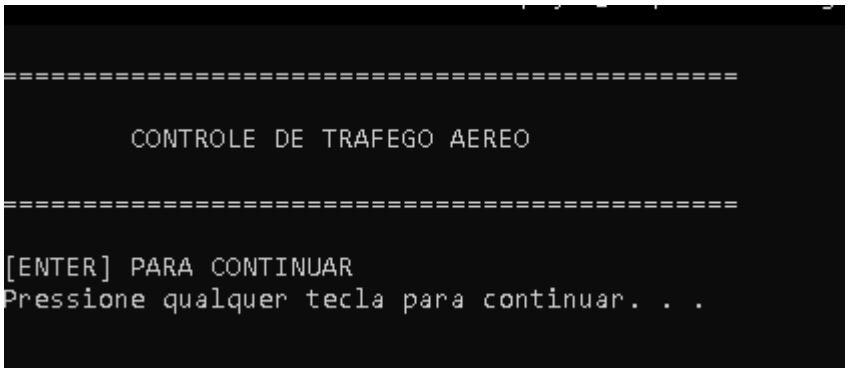
Função responsável por realizar a saída do programa, removendo os elementos das filas de aterrisagem e as de decolagem, onde esses elementos são colocados aleatoriamente nas pistas 1,2,3.

```c
int main()
{
    int i =0;
    int j =1;
    int p =0;
    int sort;
    int sort2;
    int v =1;
    int k = 0;
    int e =0;
    int prioridade;
    srand(time(NULL));

    node fila_DEC;
    node fila_AT;

    criarFila(&fila_AT);
    criarFila(&fila_DEC);


    do{
        system("cls");

        printf("\n================================================\n");
        printf("\n\tCONTROLE DE TRAFEGO AEREO\n");
        printf("\n================================================\n");

        printf("\n[ENTER] PARA CONTINUAR\n");

        system("pause");

        switch(j)
        {

        case 0:
            i = 1;
            break;

        case 1:
            sort = rand()%6;
            printf("sorte: %d\n", sort);

           for(p =0; p< sort; p++)
           {
                sort2 = rand()%2;

                if(sort2 %2 == 0)
                {
                    k = k +2;
                    //DECOLAGEM

                    inserir(&fila_DEC, k);
                    //inserirPAR;
                }
```

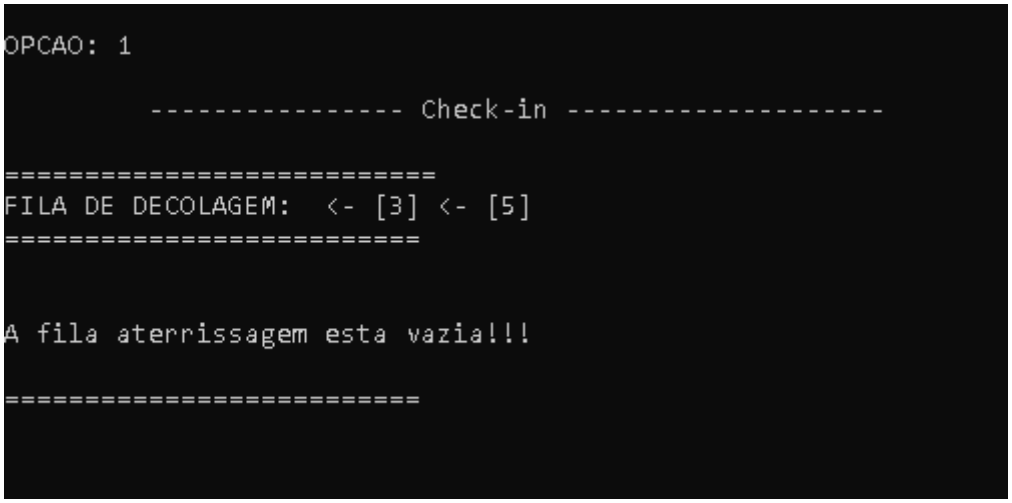
```
336                 else
337                 {
338                     v = v+2;
339                     //ATERRISSAGEM
340                     prioridade = rand()%8;
341 
342                     if(prioridade == 1)
343                     {
344                         PousoEmergencia(&fila_AT, v, prioridade);
345                         e++;
346                     //INSERIRIMPAR;
347                     }
348                     else
349                     {
350                         PousoEmergencia(&fila_AT, v, 2);
351                     }
352                 }
353             }
354             printf("\n\t ---------------- Check-in --------------------\n");
355             imprimir(fila_DEC, fila_AT);
356             printf("\n");
357             printf("\n\t ---------------- Check-out -------------------\n");
358             controlePista(&fila_AT, &fila_DEC);
359 
360             printf("\nTotal de emergencias: %d\n\n", e);
361         break;
362 
363         }
364         system("\npause");
365     }while(i!=1);
366 }
```

Função responsável por chamar todas as funções feitas nesse projeto, além de também controlar o menu para a entrada e saída de dados. O código presente no case 1 do switch é responsável pela criação dos id’s dos aviões e a inserção nas filas.

Testes realizados

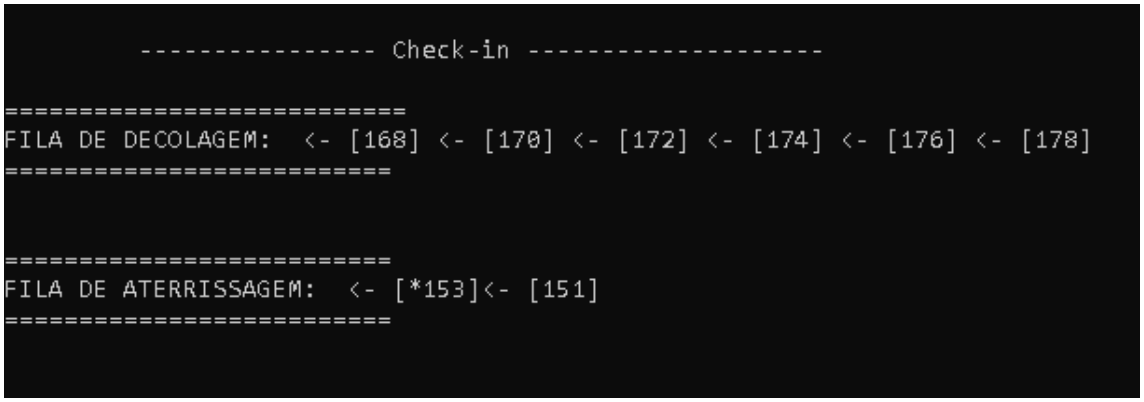


Tela Inicial do programa.



Após o usuário escolher a opção 1, foi gerado aleatoriamente dois aviões da fila de decolagem. O check-in descreve a fila inicial, sem remoções.

Em seguida, o check-out mostra a retirada dos aviões das filas, a quantidades de aviões atendidos e o total de emergências. Após apertar a tecla ENTER ou qualquer outra tecla, irá voltar ao menu inicial, e fazer os mesmos procedimentos.



Podemos observar na fila de aterrissagem a solicitação de prioridade de um avião em emergência.