Nome: João Paulo de Oliveira 11611BCC046

14° Aula prática

Uberlândia

2016

1.Código fonte:

- Main.c:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "estacionamento.h"
int main() {
    char plate[15];
    char ad ;
    int s, lane = -1, min, i ;
    while ( 1 ) {
        for ( i = 0 ; i < LANES ; i++ ) {
            printf( "lane %d: ", i ) ;
            q_display ( front[i] ) ;
            puts("");
        }
        printf( "\nArrival/Departure/Quit? ( A/D/Q ): " ) ;
        ad = getchar();
        setbuf(stdin,NULL);
        if ( toupper ( ad ) == 'Q' ) exit ( 1 ) ;

        printf ( "\nEnter license plate num:" ) ;
        gets ( plate ) ;
        ad = toupper ( ad ) ;

        if ( ad == 'A' )  { /* arrival of car */
            lane = -1 ;  /* assume no lane is available */
            min = CAPACITY ;
            for ( i = 0 ; i < LANES ; i++ ) {
                s = count ( front[i] ) ;
                if ( s < min ) {
                    min = s ;
                    lane = i ;
                }
            }
            if ( lane == -1 )
                printf ( "\nNo room available" ) ;
            else {
                insere_final( &front[ lane ], &rear[ lane ],plate ) ;
                printf ( "\npark car at lane %d slot %d\n", lane, s ) ;
            }
        }
        else {
            if ( ad == 'D' ) {  /* departure of car */
                for ( i = 0 ; i < LANES ; ++i ) {
                    s = search ( front[i], plate ) ;
                    if ( s != -1 ) {
                        lane = i ;
                        break ;
                    }
                }
                if ( i == LANES )
                    printf ( "\nno such car!!\n" ) ;
                else  {
                    printf ( "\ncar found at lane %d slot %d\n", lane, s ) ;
                    del_dq ( &front[ lane ], &rear[ lane ], s ) ;
                }
            }
            else if( ad == 'Q')
                    exit (1) ;
        }
    }
    return 0;
}
```

- Estacionamento.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "estacionamento.h"
#include "pilha.h"
/* adds a new element at the end of queue */
int search ( struct node *q, char *p ){
    int s = -1, c = 0 ;
    while ( q != NULL ) {
        if ( strcmp ( p, q -> plate ) == 0 ) {
            s = c ;
            break ;
        }
        else {
            q = q -> link ;
            c++ ;
        }
    }
    return ( s ) ;
}
void insere_final(struct node **f, struct node **r, char *p ){
    struct node *q ;
    q = ( struct node * ) malloc ( sizeof ( struct node ) ) ;
    strcpy ( q -> plate, p ) ;
    q -> link = NULL ;
    if ( *f == NULL ){    /* if the queue is empty */
        *f = q ;
    }
    else {
        (*r) -> link = q ;
    }
    *r = q ;
}
void insere_inicio( struct node **f, struct node **r, char *p ){
    struct node *q ;
    /* create new node */
    q = ( struct node * ) malloc ( sizeof ( struct node ) ) ;
    strcpy ( q -> plate, p ) ;
    q -> link = NULL ;
    if ( *f == NULL )    /* if the queue is empty */
        *f = q ;
    else {
        q -> link = *f ;
        *f = q ;
        return ;
    }
    *r = q ;
}
int count ( struct node *q ){
    int c = 0 ;
    while ( q!= NULL ) {     /* traverse the entire linked list */

        q = q -> link ;
        c++ ;
    }
    return c ;
}
void q_display ( struct node *q ) {
    while( q != NULL )    {
        printf ( "%s ", q -> plate ) ;
        q = q -> link ;
    }
}
void del_dq ( struct node **f, struct node **r, int n ) {
    if ( *f == NULL )
```

```
        printf ( "queue is empty" ) ;
    else {
        if ( n == 0 ){
            pop(f);
        }
    }
}
```

- Tipo.h:

```
#ifndef TIPO_H_INCLUDED
#define TIPO_H_INCLUDED
#define LANES 10
struct node {
    char plate [15] ;
    struct node *link ;
} *front[LANES], *rear[LANES] ;


#endif // TIPO_H_INCLUDED
```

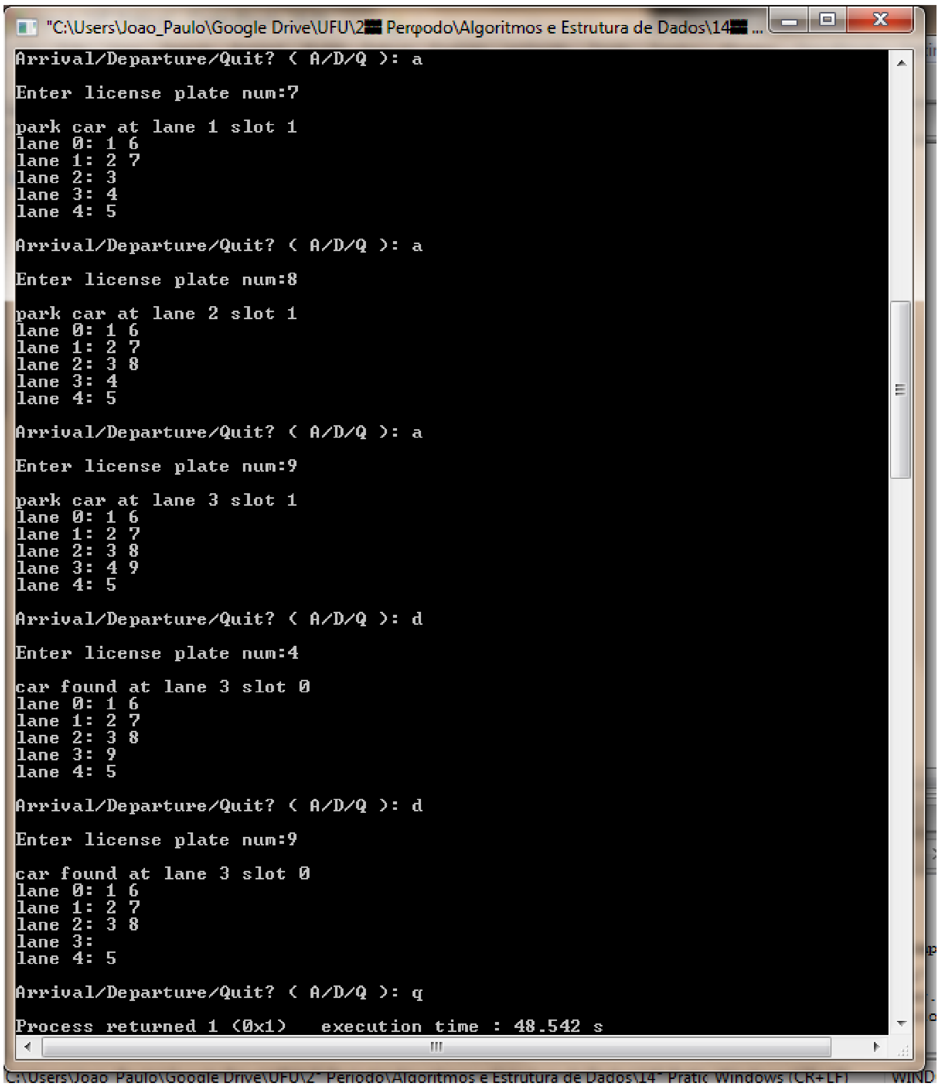
- Pilha.c:

```
#include <stdio.h>
#include "pilha.h"
#include<stdlib.h>
#include <string.h>
void push ( struct node **s, char* item ) {
    struct node *q ;
    q = ( struct node * ) malloc ( sizeof ( struct node ) ) ;
    strcpy ( q -> plate, item) ;
    q -> link = NULL ;
    *s = q ;
}

/* removes an element from top of stack */
void pop ( struct node **s ) {
    struct node *q ;
    /* if stack is empty */
    if ( *s == NULL ){
        puts("erro\n");
        return;
    }
    else {
        q = *s ;
        *s = q -> link ;
        free ( q ) ;
    }
}
```

- Pilha.h:

```
#ifndef PILHA_H_INCLUDED
#define PILHA_H_INCLUDED
#include "tipo.h"
void pop ( struct node **s );
void push ( struct node **s, char* item );

#endif // PILHA_H_INCLUDED
```

2.Print do funcionamento:


"C:\Users\Joao_Paulo\Google Drive\UFU\2▮ Perφodo\Algoritmos e Estrutura de Dados\14▮ ...
C:\Users\Joao_Paulo\Google Drive\UFU\2° Periodo\Algoritmos e Estrutura de Dados\14° Pratic Windows (CR+LF) WIND

```
Arrival/Departure/Quit? ( A/D/Q ): a

Enter license plate num:7

park car at lane 1 slot 1
lane 0: 1 6
lane 1: 2 7
lane 2: 3
lane 3: 4
lane 4: 5

Arrival/Departure/Quit? ( A/D/Q ): a

Enter license plate num:8

park car at lane 2 slot 1
lane 0: 1 6
lane 1: 2 7
lane 2: 3 8
lane 3: 4
lane 4: 5

Arrival/Departure/Quit? ( A/D/Q ): a

Enter license plate num:9

park car at lane 3 slot 1
lane 0: 1 6
lane 1: 2 7
lane 2: 3 8
lane 3: 4 9
lane 4: 5

Arrival/Departure/Quit? ( A/D/Q ): d

Enter license plate num:4

car found at lane 3 slot 0
lane 0: 1 6
lane 1: 2 7
lane 2: 3 8
lane 3: 9
lane 4: 5

Arrival/Departure/Quit? ( A/D/Q ): d

Enter license plate num:9

car found at lane 3 slot 0
lane 0: 1 6
lane 1: 2 7
lane 2: 3 8
lane 3:
lane 4: 5

Arrival/Departure/Quit? ( A/D/Q ): q

Process returned 1 (0x1)   execution time : 48.542 s
```