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:

