philosophers.c

main:
init_mutex( ): cria os mutexes para evitar data race
error_check( ): verifica erros de entrada
data.start_dinner: pega o tempo atual
start_struct( ):
create_philo( ):
cria a thread do monitor, com a função **died( )**
executa join na thread monitor
faz um while para executar join em cada philo
pausa para finalizar rotinas antes de destruir mutexes
destroy_mutex( ): destrói as mutexes criadas
libera as structs alocadas

start_struct( )

armazena dados de entrada na struct
aloca memória para philos e garfos
chama a philo_info

philo_info( )

realiza while para criação de cada philo na struct
insere os valores de cada philo (id, left_fork, right_fork etc)
cria um mutex para o garfo de cada philo

create_philo( )

realiza um while para criar a thread de cada philo
cada thread terá acesso à função routine( )

routine( )

passa a struct para a variável philo
se houver apenas 1 philo, chama a função one_philo
um while é iniciado para gerar a rotina dos philos
se o philo_id é par, espera alguns ms para os philos ímpares pegarem os garfos deles e os garfos pares
chama a função eat( )
depois de comer, o philo dorme: printa "está dormindo"
dá uma pausa para executar que o philo está dormindo
depois de dormir, o philo pensa: printa "está pensando"
verifica se algum philo morreu, nesse caso encerra o programa

one_philo( )

cria um mutex_lock para o garfo do philo
pega o tempo atual e passa para a variável da última vez que o philo comeu
printa que pegou o garfo e printa que morreu
remove o mutex_lock do garfo
passa o valor 1 para a variável data_checker

eat( )

cria um mutex_lock para os garfos do philo
seta philo->eating para 1 (philo está comendo)
atualiza o tempo da última refeição
printa que philo pegou os garfos
printa que o philo está comendo
dá uma pausa para executar que o philo está comendo
atualiza philo->had_dinner (quantas vezes ele comeu)
remove o mutex dos garfos

died( )

repassa a struct data para a função died( )
inicia um while para verificar cada philo
verifica se algum philo morreu, nesse caso encerra o programa
verifica se cada philo comeu o tanto que deveria
verifica se todos os philos comeram o que deveriam, nesse caso encerra o programa
zera o contador para começar a verificar o primeiro philo novamente