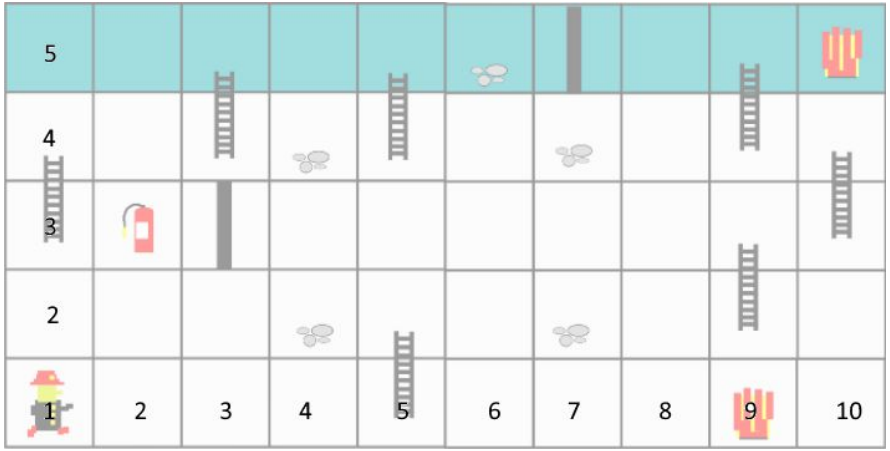
Agente Bombeiro

Leandro Novak	586927
Guilherme Neves	586986
Gustavo Bastos	551597

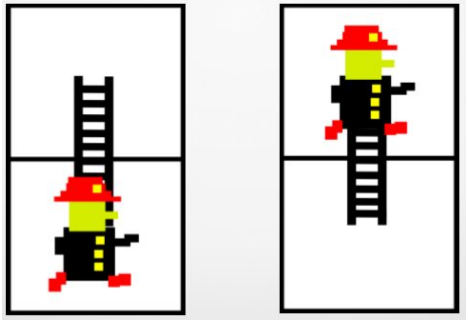
Como tudo começa!



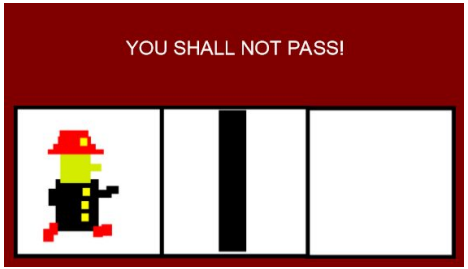

```
% Define a posição inicial do bombeiro
conteudo(1,1,bombeiro).
% Define os objetos e sua posições
conteudo(5,1,escada_inferior).
conteudo(9,1,incendio).
conteudo(4,2,entulho).
conteudo(5,2,escada_superior).
conteudo(7,2,entulho).
conteudo(9,2,escada_inferior).
conteudo(1,3,escada_inferior).
conteudo(2,3,extintor).
conteudo(3,3,parede).
conteudo(9,3,escada_superior).
conteudo(10,3,escada_inferior).
conteudo(1,4,escada_superior).
conteudo(3,4,escada_inferior).
conteudo(4,4,entulho).
conteudo(5,4,escada_inferior).
conteudo(7,4,entulho).
conteudo(9,4,escada_inferior).
conteudo(10,4,escada_superior).
conteudo(3,5,escada_superior).
conteudo(5,5,escada_superior).
conteudo(6,5,entulho).
conteudo(7,5,parede).
conteudo(9,5,escada_superior).
conteudo(10,5,incendio).
```

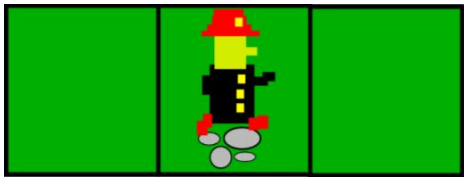
Ações: Subir e descer escadas

```
% Sobe escada
permitido(X,Y,cima) :- conteudo(X,Y,escada_inferior), Y2 is Y+1,
    conteudo(X,Y2,escada_superior).

% Desce escada
permitido(X,Y,baixo) :- conteudo(X,Y,escada_superior), Y2 is Y-1,
    conteudo(X,Y2,escada_inferior).
```



Ações: Passar por objetos



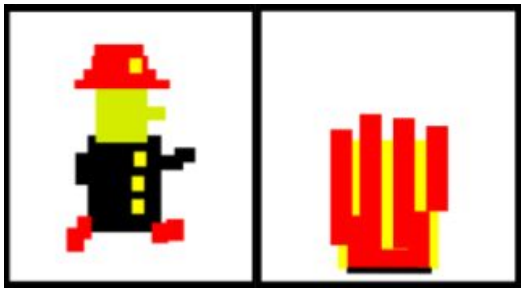


```
% Verifica parede ou incêndio
permitido(X,Y,direita) :- X1 is X+1, not(conteudo(X1,Y,entulho)),
    not(conteudo(X1,Y,parede)), not(conteudo(X1,Y,incendio)), X < 10.
permitido(X,Y,esquerda) :- X1 is X-1, not(conteudo(X1,Y,entulho)),
    not(conteudo(X1,Y,parede)), not(conteudo(X1,Y,incendio)), X > 1.
```

```
% Anda sobre entulho
permitido(X,Y,direita) :- X1 is X+1, conteudo(X1,Y,entulho),
    not(conteudo(X,Y,_)), X2 is X+2, not(conteudo(X2,Y,_)).
permitido(X,Y,esquerda) :- X1 is X-1, conteudo(X1,Y,entulho),
    not(conteudo(X,Y,_)), X2 is X-2, not(conteudo(X2,Y,_)).
```

Um caso especial, atravessando incêndios!

```
% Passa por incêndio se possuir carga no extintor
permitido(X,Y,direita) :- X1 is X+1, conteudo(X1,Y,incendio),
    carga_extintor(Carga), Carga > 0.
permitido(X,Y,esquerda) :- X1 is X-1, conteudo(X1,Y,incendio),
    carga_extintor(Carga), Carga > 0.
```



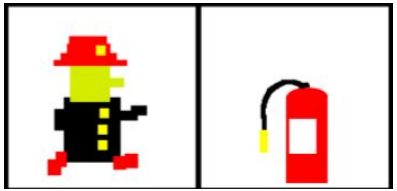
Busca em Largura

```
solucao_bl(Inicial,Item,Solucao) :- bl([[Inicial]],Solucao,Item).

% Se o primeiro estado de F for meta, então o retorna com o caminho
bl([[Estado|Caminho]|_],[Estado|Caminho],Item) :- meta(Estado,Item).

% Falha ao encontrar a meta, então estende o primeiro estado até seus sucessores
% e os coloca no final da lista de fronteira
bl([Primeiro|Outros], Solucao, Item) :-
    estende(Primeiro,Sucessores),
    concatena(Outros,Sucessores,NovaFronteira),
    bl(NovaFronteira,Solucao, Item).
```

Regras de sucessor

```
% Metodo que faz a extensao do caminho até os nós filhos do estado
estende([Estado|Caminho],ListaSucessores):-
    bagof([Sucessor,Estado|Caminho], (s(Estado,Sucessor),
    not(pertence(Sucessor,[Estado|Caminho]))),
    ListaSucessores),!.

% Se o estado não tiver sucessor, falha e não procura mais (corte)
estende( _ ,[]).

s([X,Y],[X,Y2]) :- permitido(X,Y,cima), Y2 is Y+1.
s([X,Y],[X,Y2]) :- permitido(X,Y,baixo), Y2 is Y-1.
s([X,Y],[X2,Y]) :- permitido(X,Y,direita), X2 is X+1.
s([X,Y],[X2,Y]) :- permitido(X,Y,esquerda), X2 is X-1.
```

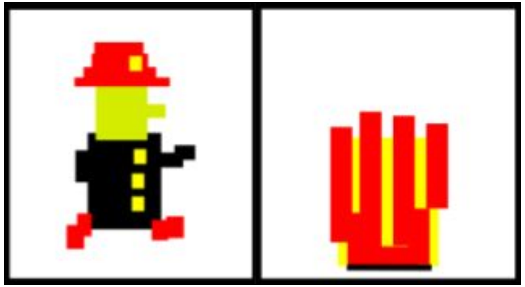
Em busca do extintor perdido!



```
% Extintor ainda possui carga, então não efetua a busca
busca_extintor([X,Y,Caminho],[X,Y,Caminho]) :-
    carga_extintor(Carga),
    Carga > 0,!.

% Extintor vazio, inicia busca por um novo extintor
busca_extintor([X,Y,Caminho],[X2,Y2,[[X2,Y2]|Caminho2]]) :-
    solucao_bl([X,Y],extintor,C),
    concatena(C,Caminho,[[X2,Y2]|Caminho2]),
    atualiza_extintor(2),
    retract(conteudo(X2,Y2,extintor)),!.

% Atualiza a carga do extintor
atualiza_extintor(Carga) :-
    retractall(carga_extintor(_)),
    assert(carga_extintor(Carga)).
```

```
% Sem incêndios para apagar
busca_incendio([X,Y,Caminho], [X,Y,Caminho]) :-
    aggregate_all(count, conteudo(_,_,incendio), Count),
    Count == 0,!.

% Tenta encontrar um incêndio e apagá-lo
busca_incendio([X,Y,Caminho], [X2,Y2,[[X2,Y2]|Caminho2]]) :-
    solucao_bl([X,Y],incendio,C),
    carga_extintor(Carga),
    NovaCarga is Carga-1,
    atualiza_extintor(NovaCarga),
    concatena(C,Caminho,[[X2,Y2]|Caminho2]),
    retract(conteudo(X2,Y2,incendio)),!.
```

Apagando Incêndios!



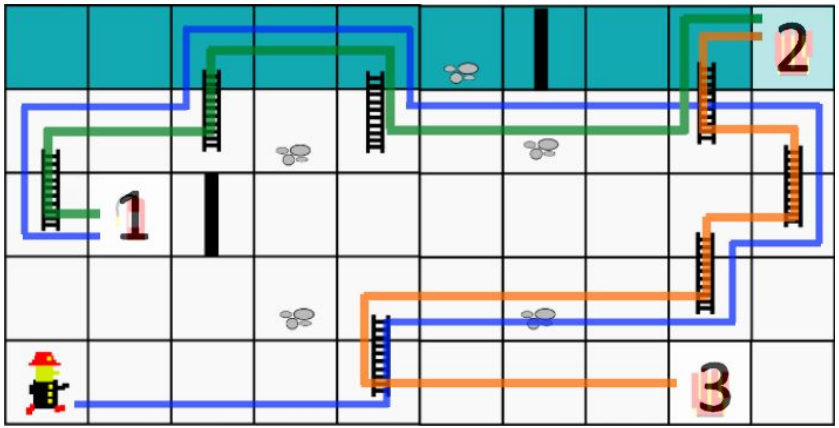
```prolog
% Sem incêndios para apagar
apaga_incendios([X,Y,Caminho], [X,Y,Caminho]) :-
    aggregate_all(count, conteudo(_,_,incendio), Count),
    Count == 0,!.

% Busca extintor e apaga até dois incêndios
apaga_incendios([X,Y,Caminho], [XF,YF,CaminhoF]) :-
    busca_extintor([X,Y,Caminho],[X2,Y2,Caminho2]),
    carga_extintor(Carga), Carga > 0,
    busca_incendio([X2,Y2,Caminho2],[X3,Y3,Caminho3]),
    busca_incendio([X3,Y3,Caminho3],[X4,Y4,Caminho4]),
    apaga_incendios([X4,Y4,Caminho4], [XF,YF,CaminhoF]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
apaga_todos_os_incendios(Arquivo, Caminho) :-
    carrega_ambiente(Arquivo),
    conteudo(X,Y,bombeiro),
    retract(conteudo(X,Y,bombeiro)),
    apaga_incendios([X,Y,[]],[_,_,CaminhoInv]),
    inverter(CaminhoInv,Caminho,[]).
```

Funcionalidades úteis!

```
carrega_ambiente(File) :-
    retractall(conteudo(_,_,_)),
    retractall(carga_extintor(_)),
    assert(carga_extintor(0)),
    set_prolog_flag(answer_write_options,[max_depth(0)]),
    open(File, read, Stream),
    call_cleanup(carrega_ambiente(Stream, _, _),
    close(Stream)).

carrega_ambiente(_, [], T) :- T == end_of_file, !.

carrega_ambiente(Stream, [T|X], _) :-
    read(Stream, T),
    assert(T),
    carrega_ambiente(Stream,X,T).
```

```
pertence(Elem,[Elem|_ ]).
pertence(Elem,[ _| Cauda]) :-
    pertence(Elem,Cauda).

concatena([ ],L,L).
concatena([Cab|Cauda],L2,[Cab|Resultado]) :-
    concatena(Cauda,L2,Resultado).

inverter([],L,L).
inverter([Cab|Cauda],L2,Aux) :-
    inverter(Cauda,L2,[Cab|Aux]).
```

Exemplo!




```
novak:src$ swipl firefighter.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- apaga_todos_os_incendios('ambiente1.pl', Caminho).
Caminho = [[1,1],[2,1],[3,1],[4,1],[5,1],[5,2],[6,2],[7,2],[8,2],[9,2],[9,3],[10,3],[10,4],
[9,4],[8,4],[7,4],[6,4],[5,4],[5,5],[4,5],[3,5],[3,4],[2,4],[1,4],[1,3],[2,3],[2,3],[1,3],[
1,4],[2,4],[3,4],[3,5],[4,5],[5,5],[5,4],[6,4],[7,4],[8,4],[9,4],[9,5],[10,5],[10,5],[9,5],
[9,4],[10,4],[10,3],[9,3],[9,2],[8,2],[7,2],[6,2],[5,2],[5,1],[6,1],[7,1],[8,1],[9,1]].

?-
```

Dúvidas?

Referências

- Trabalho 1 - Agente Bombeiro. Imagens e exemplos. Material disponibilizado pelo professor Dr. Murilo Naldi.
- Aula 5 – Prolog (Parte 2). Algoritmos de manipulação de listas. Material disponibilizado pelo professor Dr. Murilo Naldi.
- Aula 06 – Resolução de problemas. Algoritmo de Busca em Largura. Material disponibilizado pelo professor Dr. Murilo Naldi.
- Prolog count the number of times a predicate is true. Disponível em: <https://stackoverflow.com/questions/6060268/prolog-count-the-number-of-times-a-predicate-is-true>. Acesso em: 20 Out. 2019.
- Reading dynamic data from a file. Disponível em: <https://www.swi-prolog.org/FAQ/ReadDynamicFromFile.html>. Acesso em 20 Out. 2019.
- SWI-Prolog - show long list. Disponível em: <https://stackoverflow.com/questions/8231762/swi-prolog-show-long-list>. Acesso em 21 Out. 2019.
- Managing (dynamic) predicates. Disponível em: <https://www.swi-prolog.org/pldoc/man?section=dynpreds>. Acesso em 21 Out. 2019.