PFP de la fonction qui trouve les bateaux :

programme ecris en C

```
bool deja_touche (char c){
   return (c == TOUCHE || c == PLOUF);
}

void trouver_bateau(t_matrice m, int x, int y)
{
do {
x=rand ()%N;
y=rand ()%N;
}while((deja_toucher(m[x][y])));
if(m[x][y]==VIDE){
m[x][y]=PLOUF;
}
else{
m[x][y]=TOUCHE;
}
}
```

SPECIFICATION:

Entre: m[x][y]

invariant: !m[x][y] /*toutes les autres cases a part la case d'entre*/

```
while(deja_toucher){
!m[x][y]^(deja_toucher) /*Invariant^condition de boucle*/
x=rand%N; /*cors de boucle*/
y=rand%N;
m[x][y];
!m[x][y] /*Invariant*/
}
!m[x][y]^deja_toucher; /*Invariant^(negation de la condition de boucle)*/
```

Sortie: (m[x][y]=PLOUF v m[x][y]=TOUCHE)

VERIFICATION:

(1) **Entrée -> Invariant :**

m[x][y] ---> !m[x][y] (TOP)=VRAI

Justification: *en entrant de nouvelle coordonnée on remplace la valeur de la matrice à ces coordonnées par une valeur différente de ce qu'elle contenait.*

Départ : *la case contient soit (P, C, D, V, VIDE)*

Sortie : *la case contient soit (X, O).*

(2) **(Invariant condition) -> pfp("corps ∧ boucle", Invariant) :**

!m[x][y]^(deja_toucher) -> pfp("x=rand()%N, y=rand()%N",!m[x][y])

!m[x][y]^(deja_toucher) -> !m[rand()%N][rand()%N] (TOP)=VRAI

(3) **Invariant ∧ (négation de la condition de boucle) ->Sortie**

!m[x][y] ^!(deja_toucher) -> m[x][y]=O v m[x][y]=X (TOP)=VRAI

Justification: *on peut avoir soit un plouf ou un touche dans les cases non encore joués.*