

# U-16 旭川プログラミングコンテスト事前講習会

## プログラムの開発手順

旭川高専 最先端テクノロジー同好会

(リーダ木村 : 全道大会準優勝)

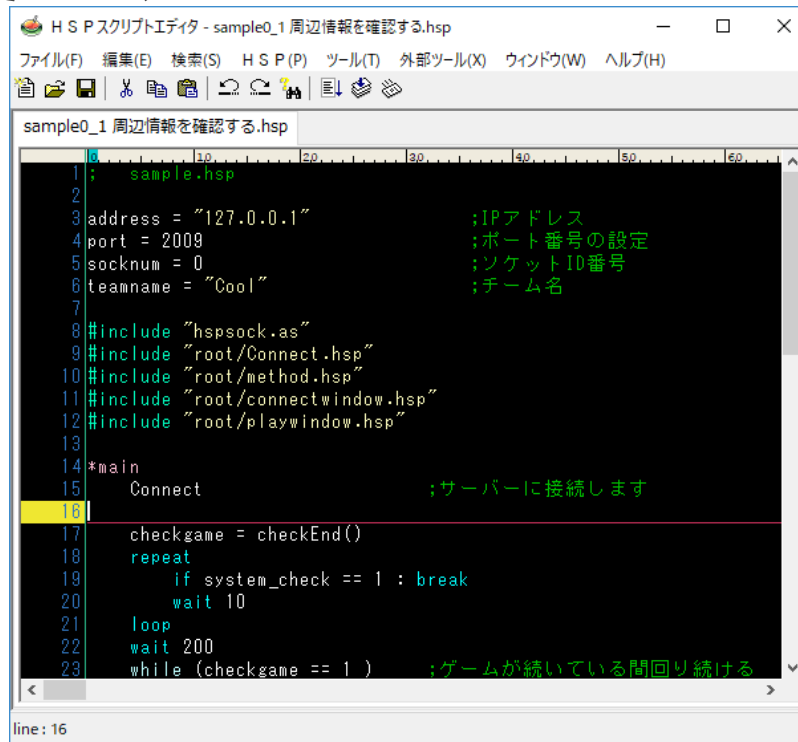
# HSPの使い方

» 今回の講習会では「HSP」  
(Hot Soup Processor)を用いて  
プログラムを書いていきます。

# フォルダの内訳

- AsahikawaProcon-Client  
HSP本体、サンプルマップ、  
クライアントプログラムが入っています
- AsahikawaProcon-Server  
サーバープログラムが入っています
- ClientManual.pdf  
クライアントプログラムのマニュアル
- ServerManual.pdf  
サーバープログラムのマニュアル

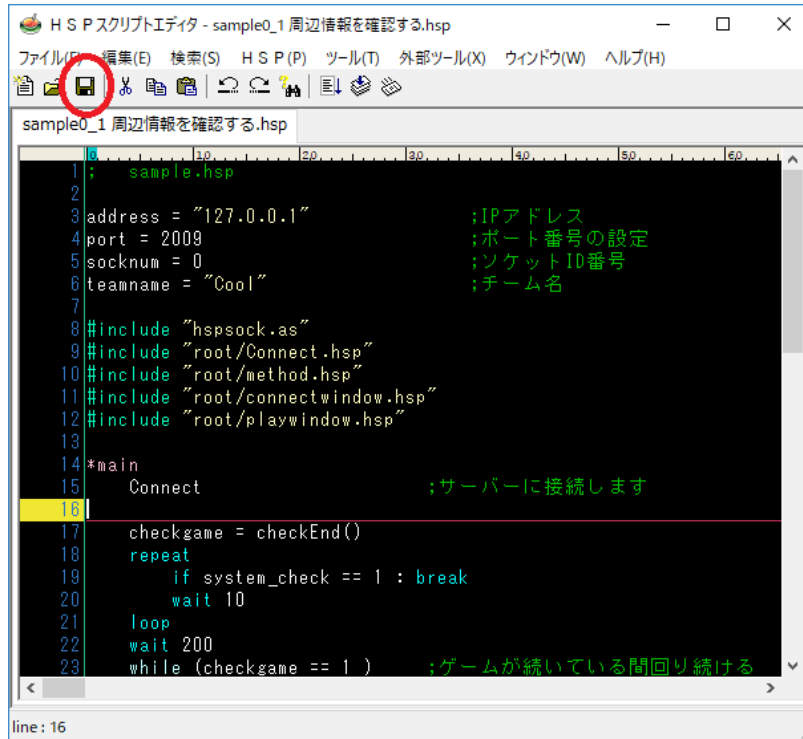
# HSPの使い方



```
sample0_1 周辺情報を確認する.hsp
1; sample.hsp
2
3address = "127.0.0.1"           ;IPアドレス
4port = 2009                    ;ポート番号の設定
5socknum = 0                    ;ソケットID番号
6teamname = "Cool"             ;チーム名
7
8#include "hspsock.as"
9#include "root/Connect.hsp"
10#include "root/method.hsp"
11#include "root/connectwindow.hsp"
12#include "root/playwindow.hsp"
13
14*main
15    Connect                    ;サーバーに接続します
16
17    checkgame = checkEnd()
18    repeat
19        if system_check == 1 : break
20        wait 10
21    loop
22    wait 200
23    while (checkgame == 1)      ;ゲームが続いている間回り続ける
```

❑ 「AsahikawaProcon-Client」 → 「HSPClient」  
→ 「sample1\_上に進む.hsp」を  
開いてみましょう。

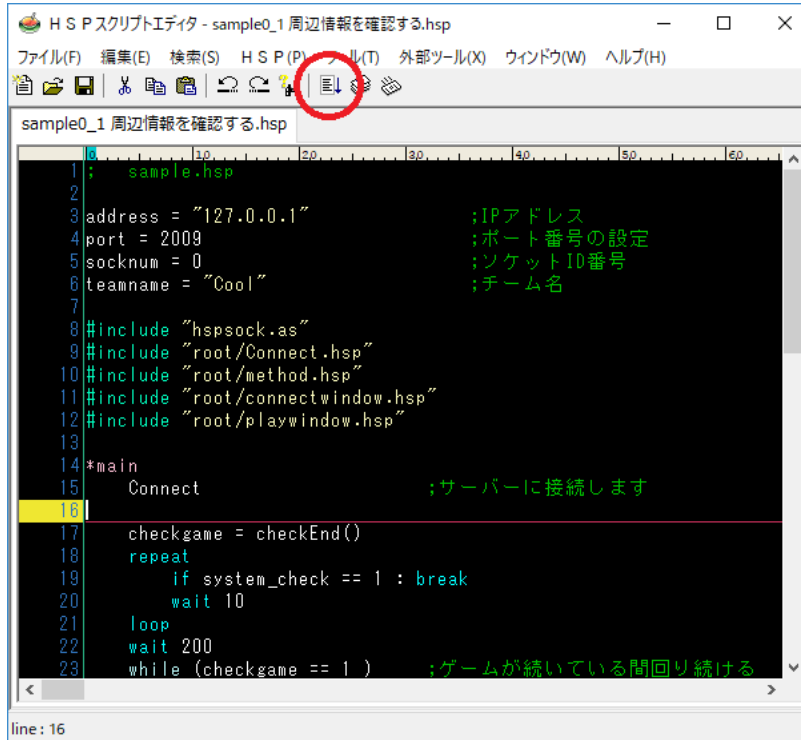
# HSPの使い方



図中の左上にある黄色のアイコン(セーブ)もしくは

「ファイル(F)」→「上書き保存(S)」で  
ファイルを保存することができます。

# HSPの使い方



```
1; sample.hsp
2
3address = "127.0.0.1"           ;IPアドレス
4port = 2009                   ;ポート番号の設定
5socknum = 0                   ;ソケットID番号
6teamname = "Cool"            ;チーム名
7
8#include "hspsock.as"
9#include "root/Connect.hsp"
10#include "root/method.hsp"
11#include "root/connectwindow.hsp"
12#include "root/playwindow.hsp"
13
14*main
15    Connect                   ;サーバーに接続します
16
17    checkgame = checkEnd()
18    repeat
19        if system_check == 1 : break
20        wait 10
21    loop
22    wait 200
23    while (checkgame == 1)     ;ゲームが続いている間繰り返し続ける
```

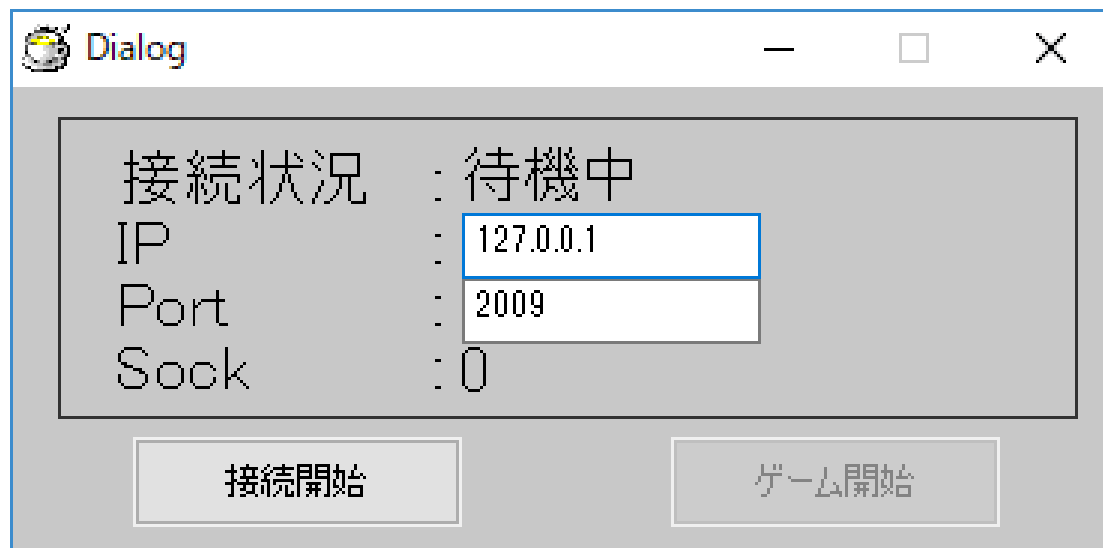


図中の下向きの青矢印のアイコン(HSP実行)もしくは

「HSP(P)」 → 「コンパイル+実行(F5)」を選択すると

クライアントプログラムが実行されます

# HSPの使い方



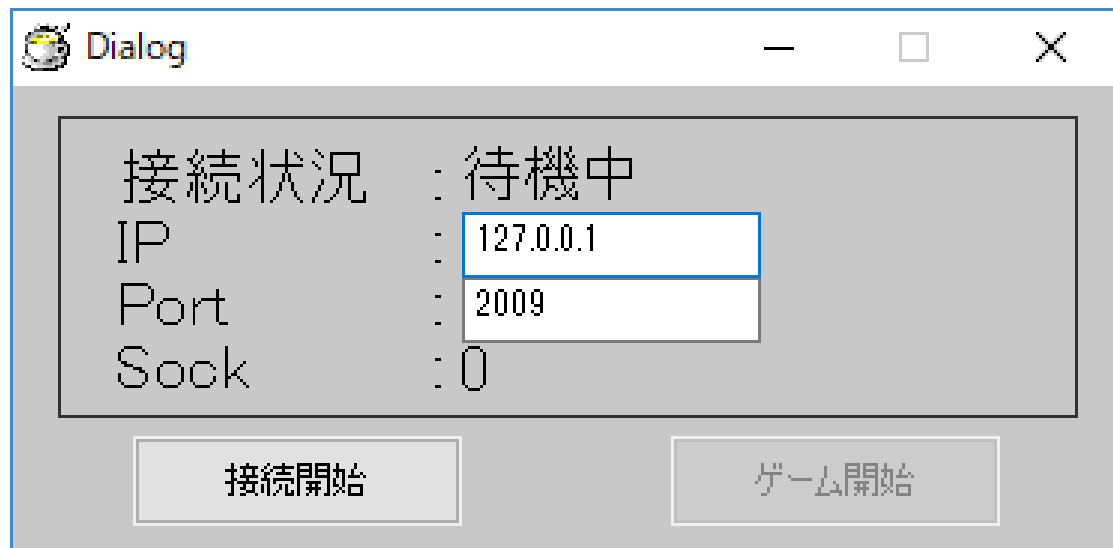
- クライアントプログラムでは、プログラムを実行すると  
このようなウィンドウが表示されます

# サーバープログラムの使い方

» 対戦で用いられる  
サーバープログラムの使い方です。



# サーバープログラムの使い方



- まず、クライアントプログラムを実行して出てきた画面の「接続開始」のボタンを1回だけ押します。

# サーバープログラムの使い方



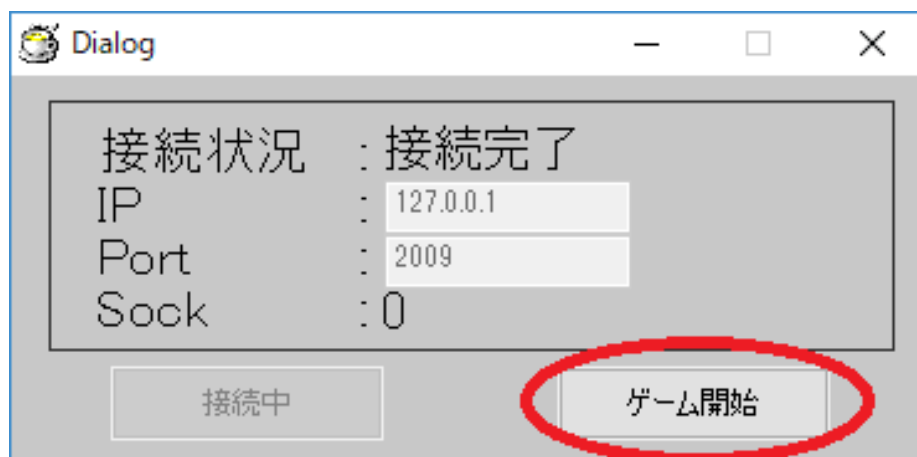
- ❓ 「AsahikawaProcon-Server」 → 「Windows」  
→ 「AsahikawaProcon-Server.exe」  
を開いてみましょう。

# サーバープログラムの使い方



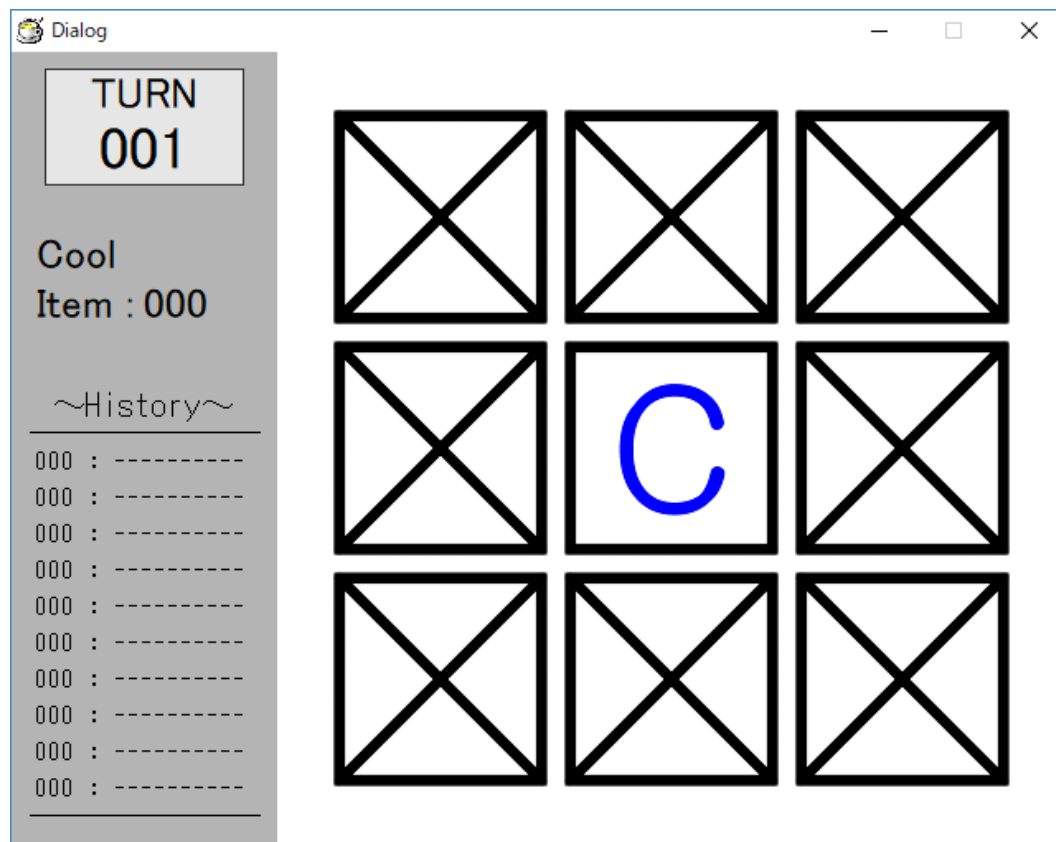
- 2つの「待機開始」ボタンのうち、ポート番号が「2009」の方のボタンを押します。

# サーバープログラムの使い方



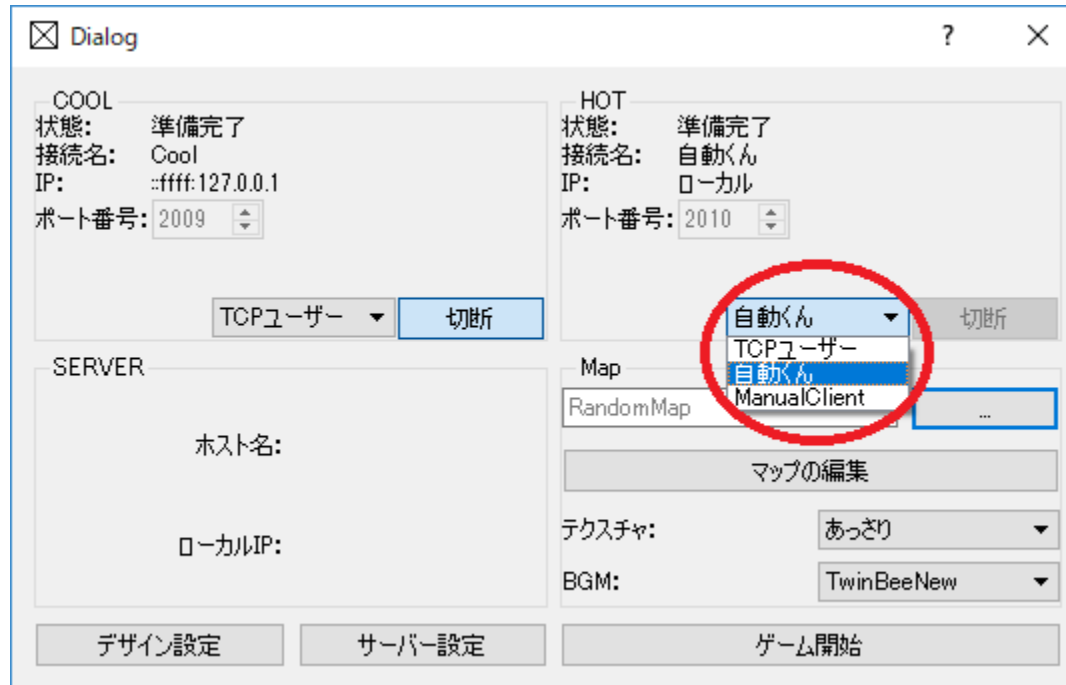
- ❑ クライアントにて左のようなダイアログが表示されるので「OK」を押し、その後、クライアントの「ゲーム開始」を押します。

# サーバープログラムの使い方



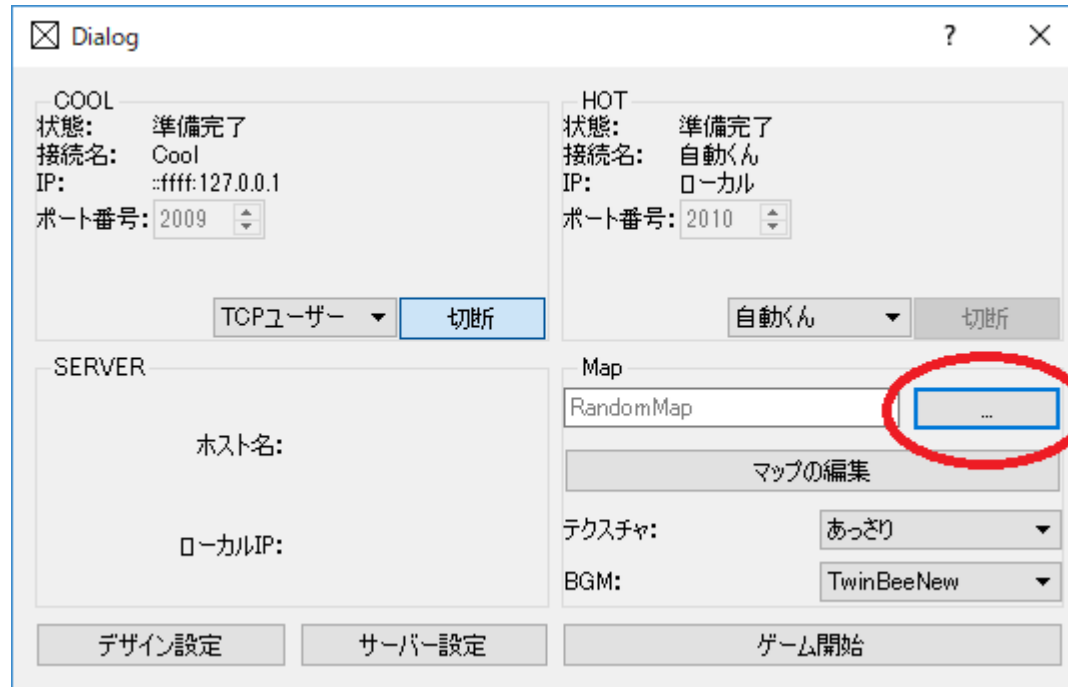
- このような画面が現れるので、クライアントは一旦この状態で待機します。

# サーバープログラムの使い方



- ❓ ポート番号が「2010」の方の「TCPユーザー」と書かれているボタンをクリックし、「自動くん」に変更します。

# サーバープログラムの使い方

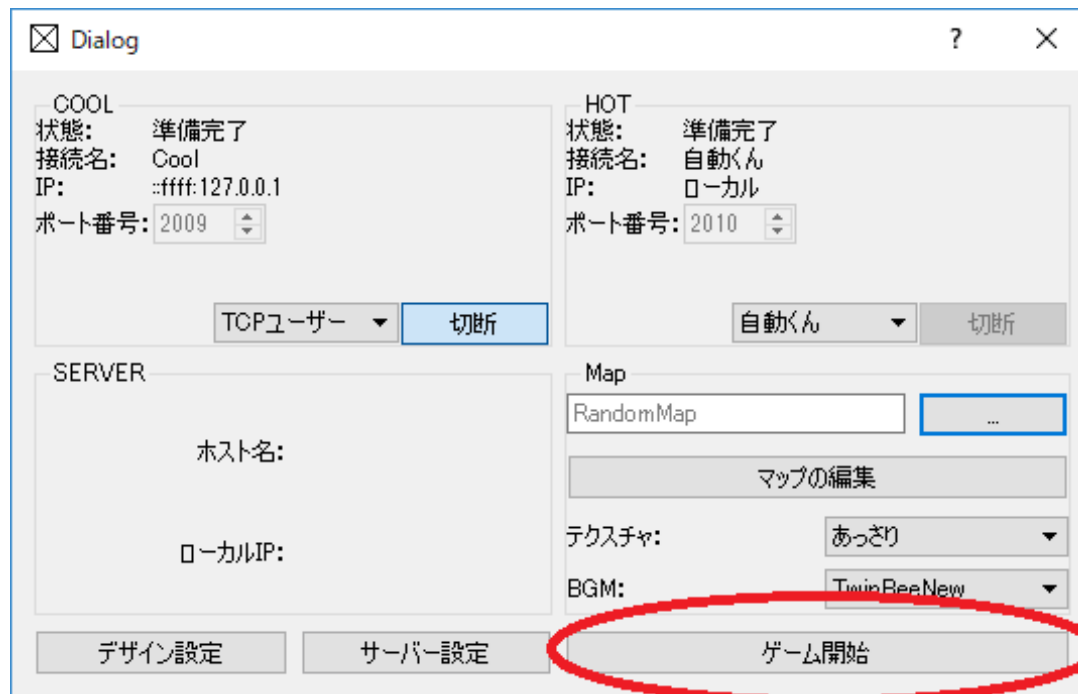


❓ ウィンドウ右側の「...」から、マップを選択します。

今回は「AsahikawaProcon-Client」

→ 「SampleMap」からサンプルマップ03を使います。

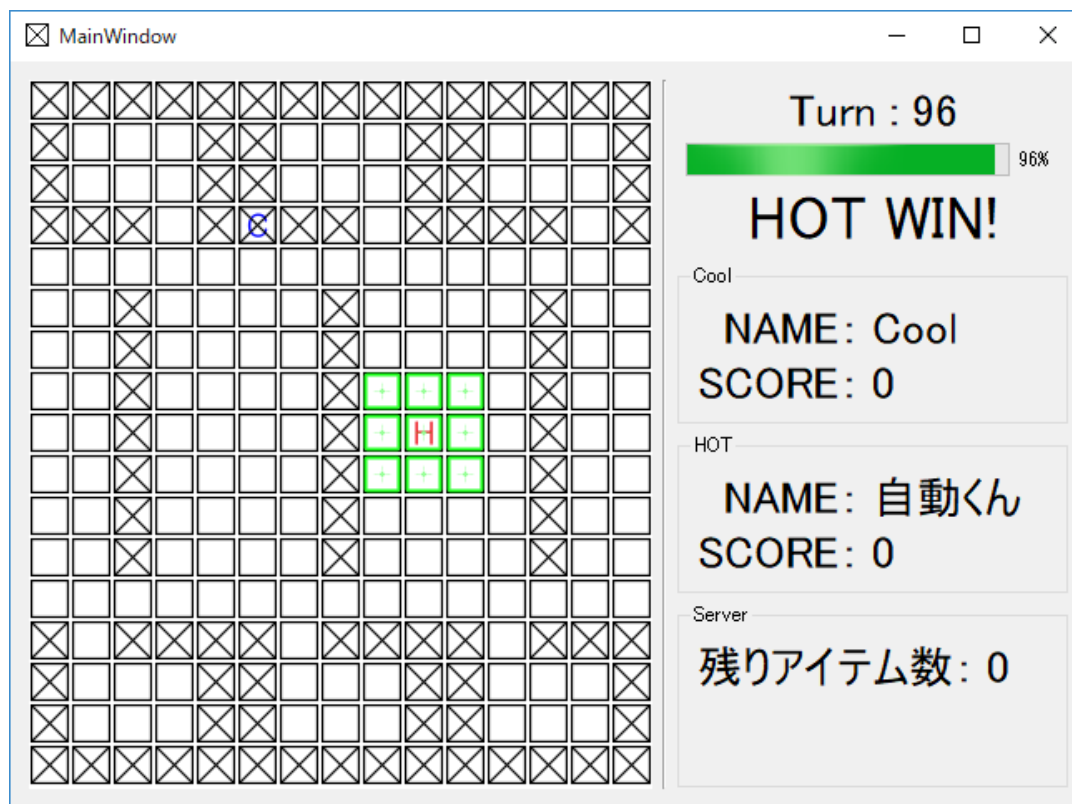
# サーバープログラムの使い方



- ❓ 右下にある「ゲーム開始」を押すと競技が始まります。



# サーバープログラムの使い方



2 以上のような画面が現れ、競技が進行していきます。

# サーバープログラムの使い方(補足)

■ 次の競技を行う場合は、一度クライアントとサーバーをウィンドウ右上の「×」ボタンを押して閉じてください。

(サーバー→クライアントの順番に閉じてください)

■ クライアント側で「ゲーム開始」を押さずにサーバーを動かしてしまうとサーバーがフリーズします。

# 競技プログラムの作成

» 競技プログラムの  
簡単な書き方です。

# 競技用プログラムの作成

■ 今回はサンプルプログラムを使用します。

- Sample1            上に進む
- Sample0\_1        周辺情報を確認する
- Sample0\_2        Lookを使う
- Sample0\_3        Searchを使う
- Sample0\_4        Putを使う
- Sample1\_2        壁を見つけたら回避する
- Sample1\_3        壁を見つけたら回避する2

# 周辺情報の確認

» Sample0\_1 を用いて、  
getReadyで受け取る周辺情報に  
何が入っているのかを確認しましょ  
う。

# 周辺情報の確認

- ❑ Sample0\_1 を使用します。
- ❑ 主なプログラムは23～36行目です。

```
23 while (checkgame == 1 )
24     getReady
25     ;デバッグログに周辺情報を記録します
26     logstring = ""
27     foreach Value
28         logstring = logstring + str(Value.cnt)
29     loop
30     logmes(logstring)
31     ;ここまで
32     ;=====
33     walkUp
34     ;=====
35     checkgame = checkEnd()
36 wend
```

ループ開始

GetReadyで周辺情報を入手

DebugWindowに周辺情報を出力

walkUpで上に向かって移動

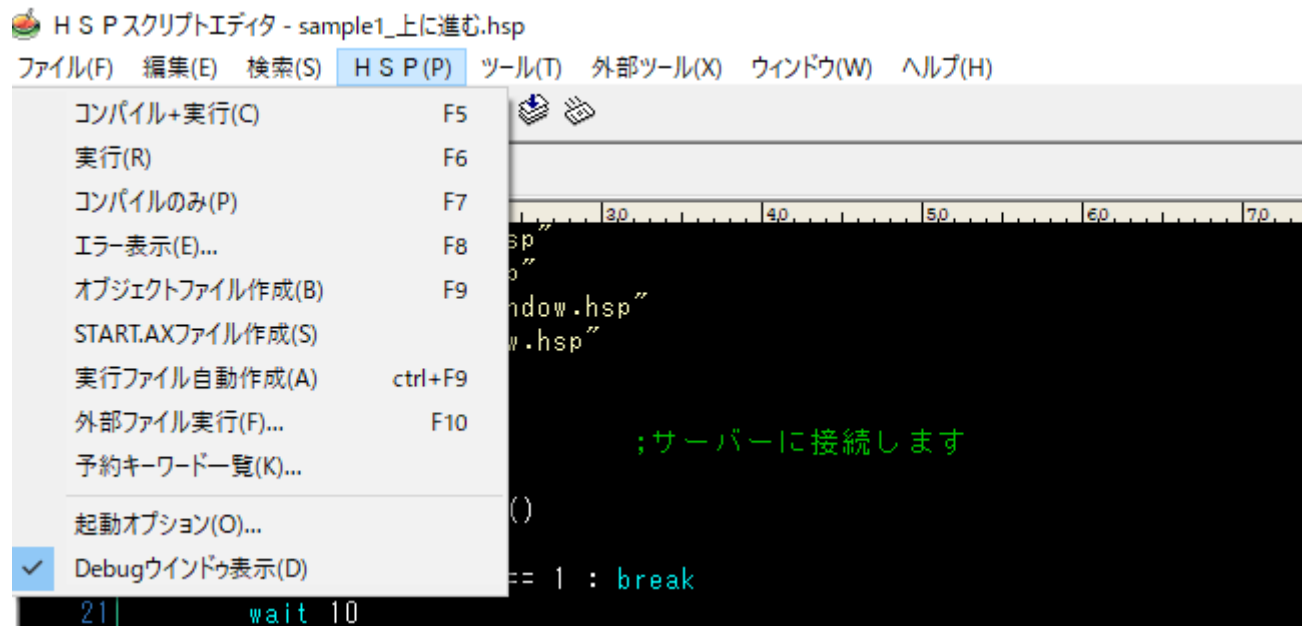
以上繰り返し

# 周辺情報の確認

```
;デバッグログに周辺情報を記録します
logstring = ""
foreach Value
    logstring = logstring + str(Value.cnt)
loop
logmes(logstring)
;ここまで
```

- 一部のサンプルには以上の処理が書かれています。  
これは後述するDebugWindowに  
周辺情報を記述するための処理です。  
コピーしてプログラムのデバッグに利用してください。

# 周辺情報の確認



- ❓ HSPにてsample0\_1を開き、「HSP(P)」  
→「Debugウィンドウ表示(D)」を押して  
チェックを入れます。

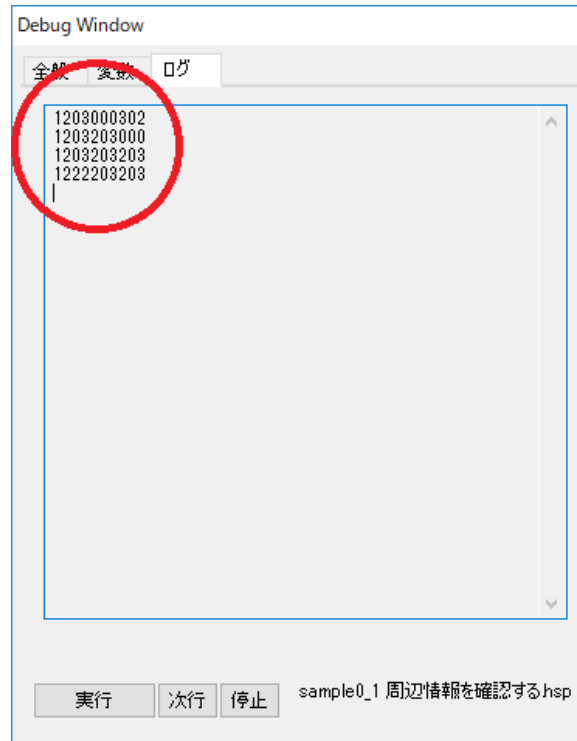


# 周辺情報の確認



- 選択した後、プログラムを実行すると「DebugWindow」という名前のウィンドウが表示されるので、ウィンドウの「ログ」を選択します。

# 周辺情報の確認




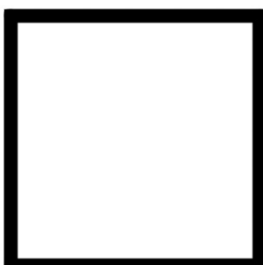
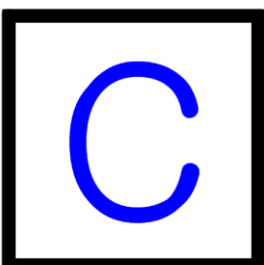
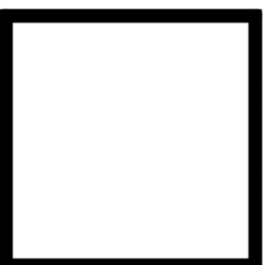

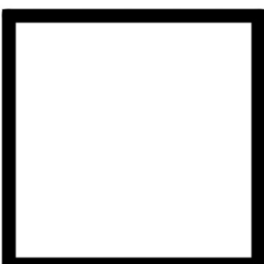



❓ プログラムを実行してみると数値が並びます。  
この数値が周辺情報です。

※サンプルマップ01を使用しています

# 周辺情報の確認

1	2	0	3	0	0	0	3	0	2
---	---	---	---	---	---	---	---	---	---

2		0		3	
0		0		0	
3		0		2	

# Look,Searchの確認

» Sample0\_2,Sample0\_3を用いて、  
LookとSearchの動きを  
確認しましょう。

# 周辺情報の確認

- ❑ Sample0\_2とSample0\_3を使用します。
- ❑ 主なプログラムは23～37行目です。

```
23  while (checkgame == 1 ) ← ;ゲームが続いている
24      getReady ←
25      ;=====
26      LookRight ← LookRight(SearchRight)で
27      ;=====
28
29      ;デバッグログに周辺情報を記録します
30      logstring = ""
31      foreach Value
32          logstring = logstring + str(Value.cnt)
33      loop
34      logmes(logstring)
35      ;ここまで
36      checkgame = checkEnd()
37  wend ← 以上繰り返し
```

ループ開始

GetReadyで周辺情報を入手

LookRight(SearchRight)で情報を取得

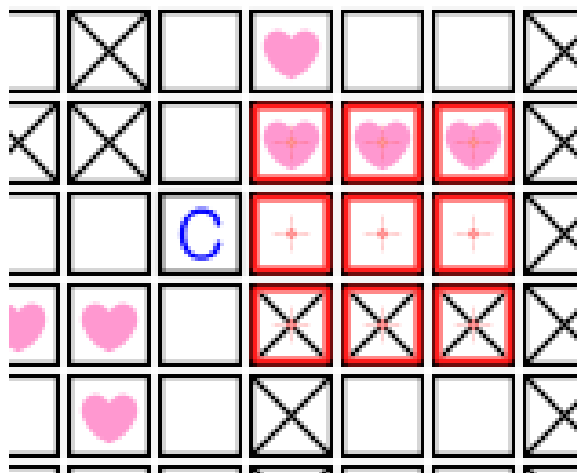
DebugWindowにLook(Search)で得た情報を出力

以上繰り返し

# Look,Searchの確認

- 先ほどと同様に、Sample0\_2とSample0\_3を動かしてみましょう。
- Sample0\_2ではサンプルマップ01、Sample0\_3ではサンプルマップ02を使いましょう。

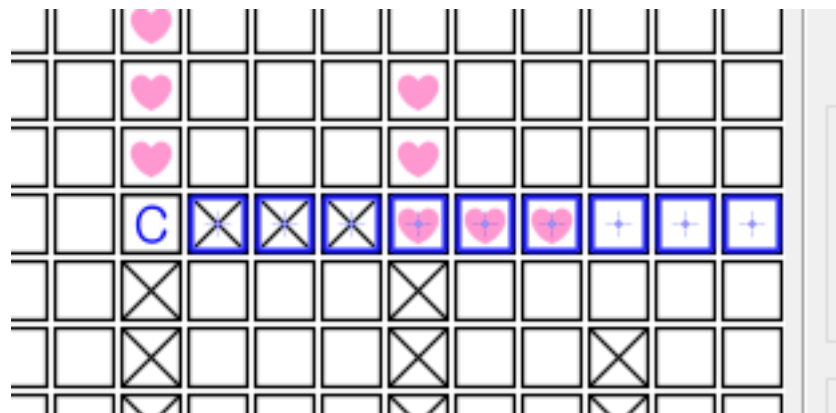
# Look,Searchの確認



? Sample0\_2でサンプルマップ01(Look)

1	3	3	3	0	0	0	3	3	3
---	---	---	---	---	---	---	---	---	---

# Look,Searchの確認



❓ Sample0\_3でサンプルマップ02(Search)

1	2	2	2	3	3	3	0	0	0
---	---	---	---	---	---	---	---	---	---



# Putの確認

» Sample0\_4を用いて、Putの動きを  
確認しましょう。

# 周辺情報の確認

- ❑ Sample0\_4を使用します。
- ❑ 主なプログラムは23～30行目です。

```
23  while (checkgame == 1 )  
24      getReady  
25      ;=====  
26      PutUp  
27      ;=====  
28  
29      checkgame = checkEnd()  
30  wend
```

ループ開始

GetReadyで周辺情報を入手

PutUpで上方向に壁を置く

以上繰り返し

# Putの確認

- Sample0\_4を実行し、  
サンプルマップ01の上で動かしてみましょう。
- putUpが実行され、
- 上方向に壁が設置されます。

# 簡単なプログラムの設計

» Sample1 を改良して、壁を見つけたら回避できるようにしよう

# 簡単なプログラムの設計

```
while (checkgame == 1 )  
  getReady  
  ;=====  
  walkUp  
  ;=====  
  checkgame = checkEnd()  
wend
```



```
getReady|  
;=====  
if (Value@.2 == 2){  
  walkLeft  
}else{  
  walkUp  
}  
;=====
```

Sample1

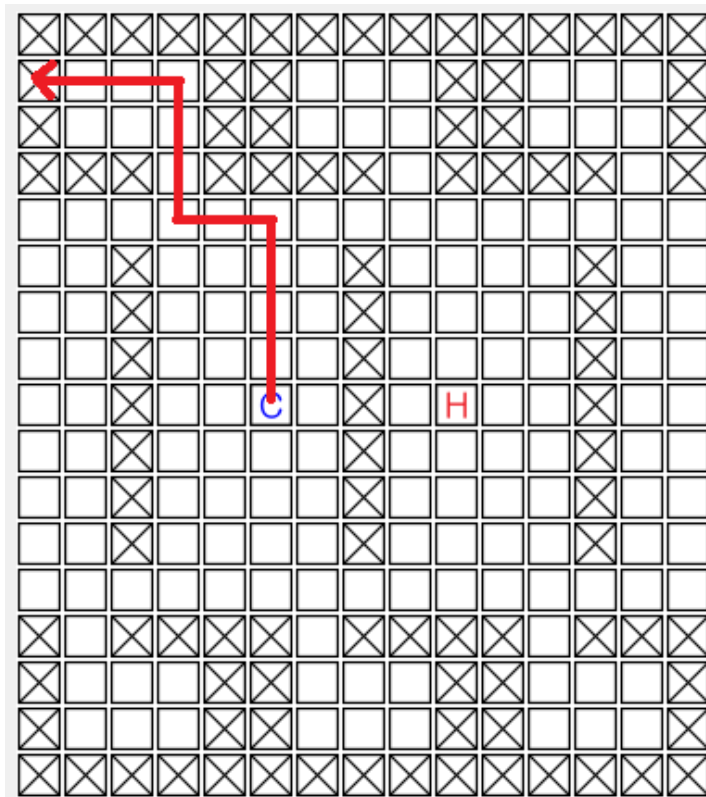
Sample1\_2

- Sample1 と Sample1\_2 の変更点は以上の通りです。

Sample1 では上のみ進みますが、Sample 1\_2 では、

上のマスに壁がある場合は左に移動します。

# 簡単なプログラムの設計



- Sample1\_2を実行して、サンプルマップ03の上で動かしてみましょう。上図のような経路を動きます。

# 簡単なプログラムの設計

```
getReady|
;=====
if(Value@.2 == 2){
    walkLeft
}else{
    walkUp
}
;=====
```

Sample1\_2



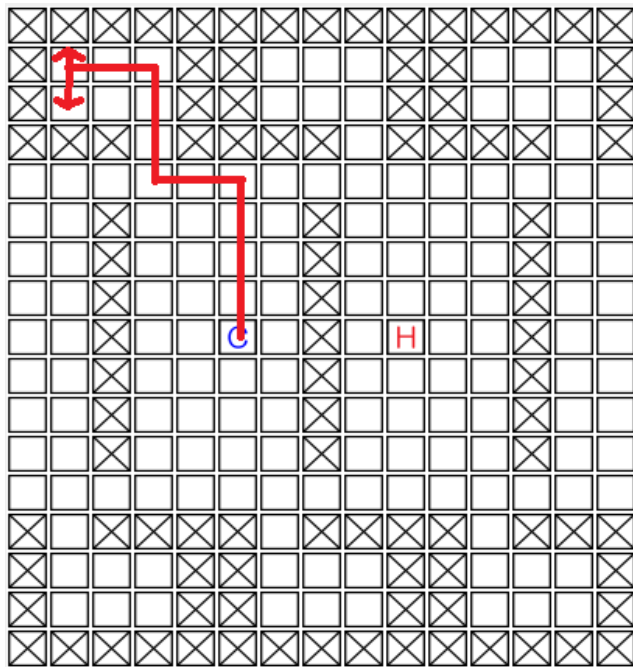
```
;=====
if(Value@.2 != 2){
    walkUp
}
else : if(Value@.4 != 2){
    walkLeft
}
else : if(Value@.8 != 2){
    walkDown
}
else : if(Value@.6 != 2){
    walkRight
}
;=====
```

Sample1\_3

- Sample1\_3はSample1\_2をさらに変えたものです。

4方向に対応し、「壁があれば避ける」から、  
「壁でなければその方向に進む」に変更されました。

# 簡単なプログラムの設計



Sample1\_3を実行して、サンプルマップ03の上  
で

動かしてみましょう。上図のような経路を動きます。

~~壁の代わりの代わりに、身動きが取れなくなりました。~~



# 簡単なプログラムの設計

- 同じ場所で何度も繰り返して動いている状態を  
**ループ**といい、このループを回避できるような  
プログラムを作らなければなりません。
- 今回の講習会でHSPを用いて、  
実際に簡単な競技用プログラムを作ってみましょ  
う。

# プログラムの開発手順は以上です

» テキストと各種サンプルを確認しながら、  
競技用プログラムを作っていきます。