全国高校生プログラミングコンテスト
CHaserOnline
ステップアップヒント２

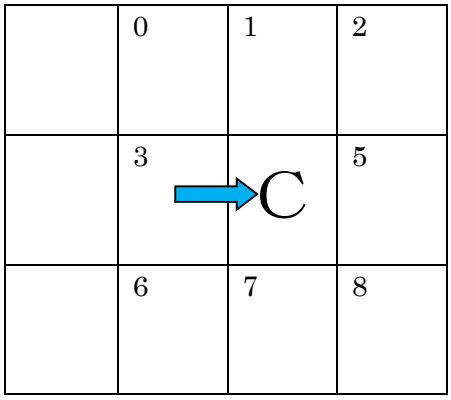
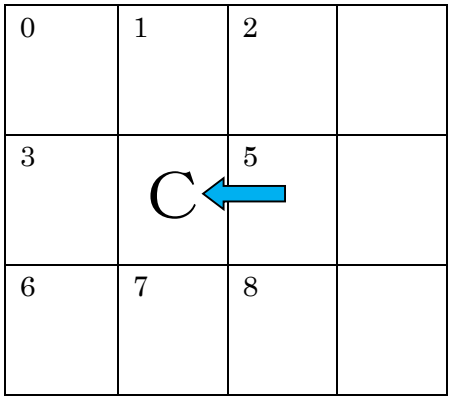
9 コマンドについて

コマンドについて説明します。CやHなどのプレイヤ（以下プレイヤ）の制御をおこなうにはコマンドを使用します。コマンドには以下のものがあります（2018.6.1 現在)。説明中のプレイヤはCと表記します。また、戻り値が入る変数は整数型の一次元配列returnNumber[10]とします。各行動のたびに動きに見合った疲労がたまる（＝減点）ため、よく考えて行動させる必要があります。

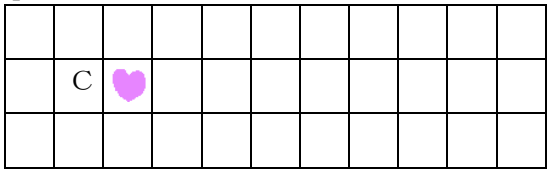
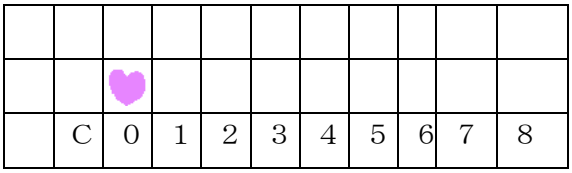
一連のコマンド発行後に「user=」メッセージを受け取った場合はゲーム終了となりますので、クライアントプログラムを終了してください。

<table>
<tr><td colspan="2">A　準備コマンド GetReadyMove（サーバに接続し、プレイヤの周囲情報を得る）</td></tr>
<tr><td>コマンド名</td><td>機能</td></tr>
<tr><td>gr
gru
grd
grl
grr</td><td>サーバに接続し、自分のターンであればプレイヤの周囲情報を得る。
gr・・・移動せずに、周囲情報を得る
gru・・・上に１マス移動して、周囲情報を得る。
grd・・・下に１マス移動して、周囲情報を得る。
grl・・・左に１マス移動して、周囲情報を得る。
grr・・・右に１マス移動して、周囲情報を得る。

<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>C</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>
マス目にある数字は returnNumber[]の添え字と一致する。
また、数字の並びは移動先の周囲 9 マスの左上が 0 となる。</td></tr>
</table>

B　動作コマンド walk 系（指定した方向へ 1 マス移動する）	
コマンド名	機能
wu wd wl wr	指定した方向へ 1 マス移動する。 wu・・・上 wd・・・下 wl・・・左 wr・・・右 ※各種のアイテム等を探しに歩くので、アイテムを獲ない限り疲労がたまっていきます。 wr の例  マス目にある数字は returnNumber[]の添え字と一致する。 また、数字の並びは wu, wd, wl の場合も移動先の周囲 9 マスの左上が 0 となる。 wl の例 

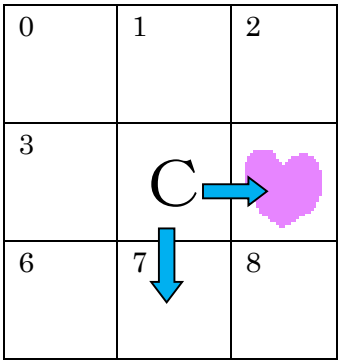
B　動作コマンド Put&Search 系
（指定した方向へアイテムを置き、進行方向右側に移動し真っすぐ 9 マスの情報を得る）

コマンド名	機能
pu3su pd3sd pl3sl pr3sr	指定した方向、真っすぐ 9 マスの情報を得る。 pu3su・・・上 pd3sd・・・下 pl3sl・・・左 pr3sr・・・右 ※アイテムを置いた分、疲労していきます。 pr2sr の動作例  ↓アイテムを置いたら ↓進行方向右側に移動 ↓真っすぐ 9 マスの情報を得る  マス目にある数字は returnNumber[]の添え字と一致する。 また、数字の並びはどの方向でも左上のマスが 0 となる。

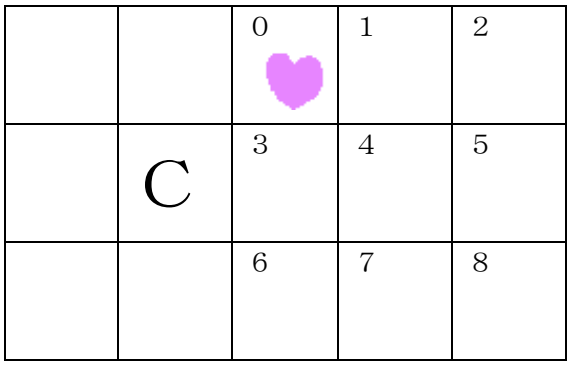
B　動作コマンド Put&Look 系
（指定した方向へアイテムを置き、進行方向右側に移動し周囲情報を得る）

コマンド名	機能
pu3lu pd3ld pl3ll pr3lr	指定した方向へアイテムを置く。 pu3lu・・・上 pd3ld・・・下 pl3ll・・・左 pr3lr・・・右 ※アイテムを置いた分、疲労していきます。

pr3lr の例



↓進行方向右側に移動して
↓周囲９マスの情報を得る

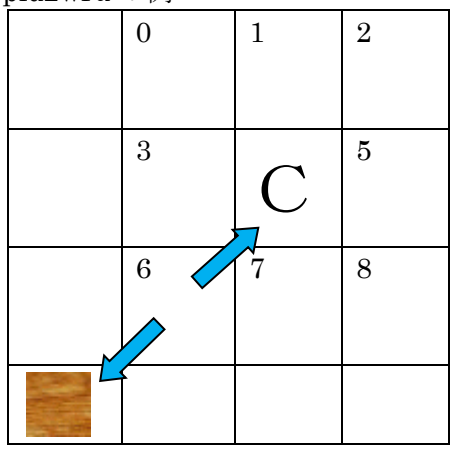


マス目にある数字は returnNumber[]の添え字と一致する。

<table>
<tr><td colspan="2">B　動作コマンド put&walk 系（指定した方向へ土を置き、逆向きに１マス移動する）</td></tr>
<tr><td>コマンド名</td><td>機能</td></tr>
<tr><td>pu2wd
pd2wu
pl2wr
pr2wl
pru2wld
plu2wrd
prd2wlu
pld2wru</td><td>指定した方向へ土を置き、逆向きに１マス移動する。

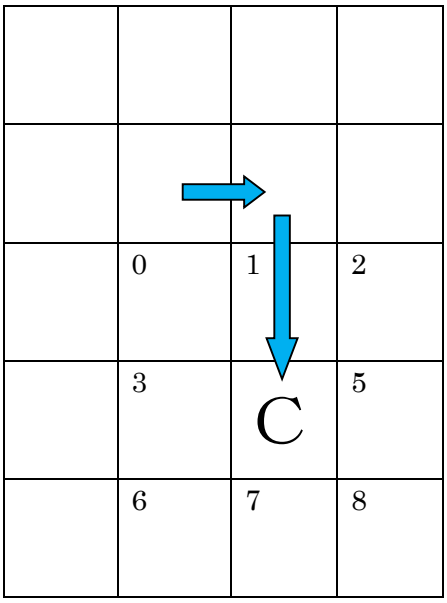
pu2wd・・・上に土を置き、下へ移動
pd2wu・・・下に土を置き、上へ移動
pl2wr・・・左に土を置き、右へ移動
pr2wl・・・右に土を置き、左へ移動
pru2wld・・・右上に土を置き、左下へ移動
plu2wrd・・・左上に土を置き、右下へ移動
prd2wlu・・・右下に土を置き、左上へ移動
pld2wru・・・左下に土を置き、右上へ移動

※土を置きなおかつ一歩下がるので、ｐｕｔよりも疲労します。しかし、土を相手に命中させたときには、かなりの得点を奪うことができます。

pld2wru の例
 

※  は土

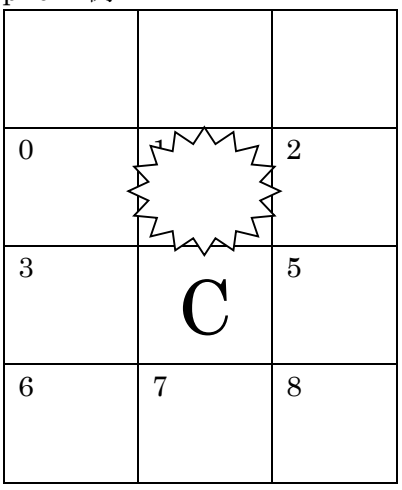
マス目にある数字は returnNumber[]の添え字と一致する。</td></tr>
</table>

B　動作コマンド　kei 系（桂馬に似た動き）	
コマンド名	機能
keiru keird keilu keild	指定した方向へ桂馬に似た動きをする。（縦２マス・横１マス） keiru・・・右上へ桂馬のような動きをする keird・・・右下へ桂馬のような動きをする keilu・・・左上へ桂馬のような動きをする keild・・・左下へ桂馬のような動きをする keird の例  マス目にある数字は returnNumber[]の添え字と一致する。

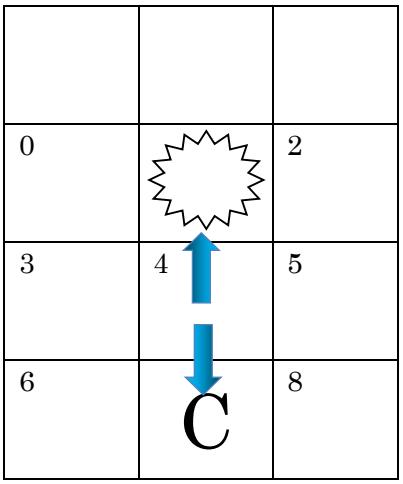
<table>
<tr><td colspan="2">B　動作コマンド put 系（砕く動作）</td></tr>
<tr><td>コマンド名</td><td>機能</td></tr>
<tr><td>pr0
pl0
pu0
pd0</td><td>指定した方向の土を砕きます。

pr0・・・右にある土を砕きます。
pl0・・・左にある土を砕きます。
pu0・・・上にある土を砕きます。
pd0・・・下にある土を砕きます。

※ 砕く動作ですので、疲労がたまりやすいですが先に進めるようになります。

pu0 の例
 

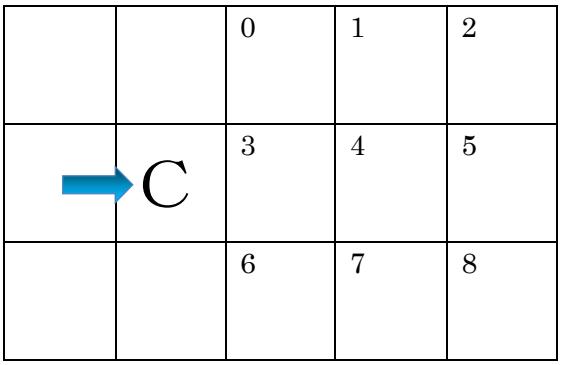
マス目にある数字は returnNumber[]の添え字と一致する。</td></tr>
</table>

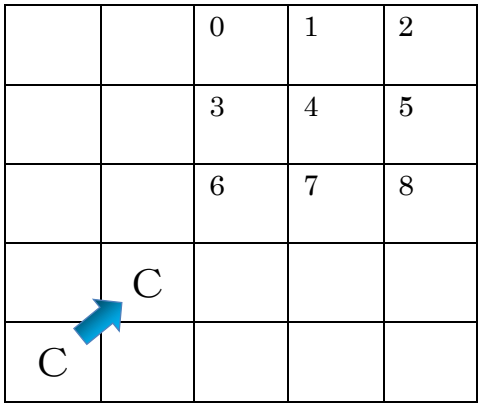
B　動作コマンド系（砕く動作）	
コマンド名	機能
pu0wd pd0wu pl0wr pr0wl pru0wld plu0wrd prd0wlu pld0wru	指定した方向の土を砕きつつ、その逆方向に移動します。 pu0wd・・・上の土を砕いて下に移動します。 pd0wu・・・下の土を砕いて上に移動します。 pl0wr・・・左の土を砕いて右に移動します。 pr0wl・・・右の土を砕いて左に移動します。 pru0wld・・右上の土を砕いて左下に移動する。 plu0wrd・・左上の土を砕いて右下に移動する。 prd0wlu・・右下の土を砕いて左上に移動する。 pld0wru・・左下の土を砕いて右上に移動する。 ※　砕く動作ですので、疲労がたまりやすいです。 pu0wd の例  マス目にある数字は returnNumber[]の添え字と一致する。

<table>
<tr><td colspan="2">B　動作コマンド dig 系（1 歩歩いて指定した方向の周囲 9 マスの情報を得る）</td></tr>
<tr><td>コマンド名</td><td>機能</td></tr>
<tr><td>du
dd
dl
dr
dru
drd
dlu
dld</td><td>指定した方向の周囲 9 マスの情報を得る。

du・・・一歩上に移動して、上を探査します。
dd・・・一歩下に移動して、下を探査します。
dl・・・一歩左に移動して、左を探査します。
dr・・・一歩右に移動して、右を探査します。
dru・・一歩右上に移動して、右上を探査します。
drd・・一歩右下に移動して、右下を探査します。
dlu・・一歩左上に移動して、左上を探査します。
dld・・一歩左上に移動して、左上を探査します。

※一歩歩いて見えない位置を広く見るので、だいぶ疲労します。

dr の例
 

dru の例
 

マス目にある数字は returnNumber[]の添え字と一致する。</td></tr>
</table>

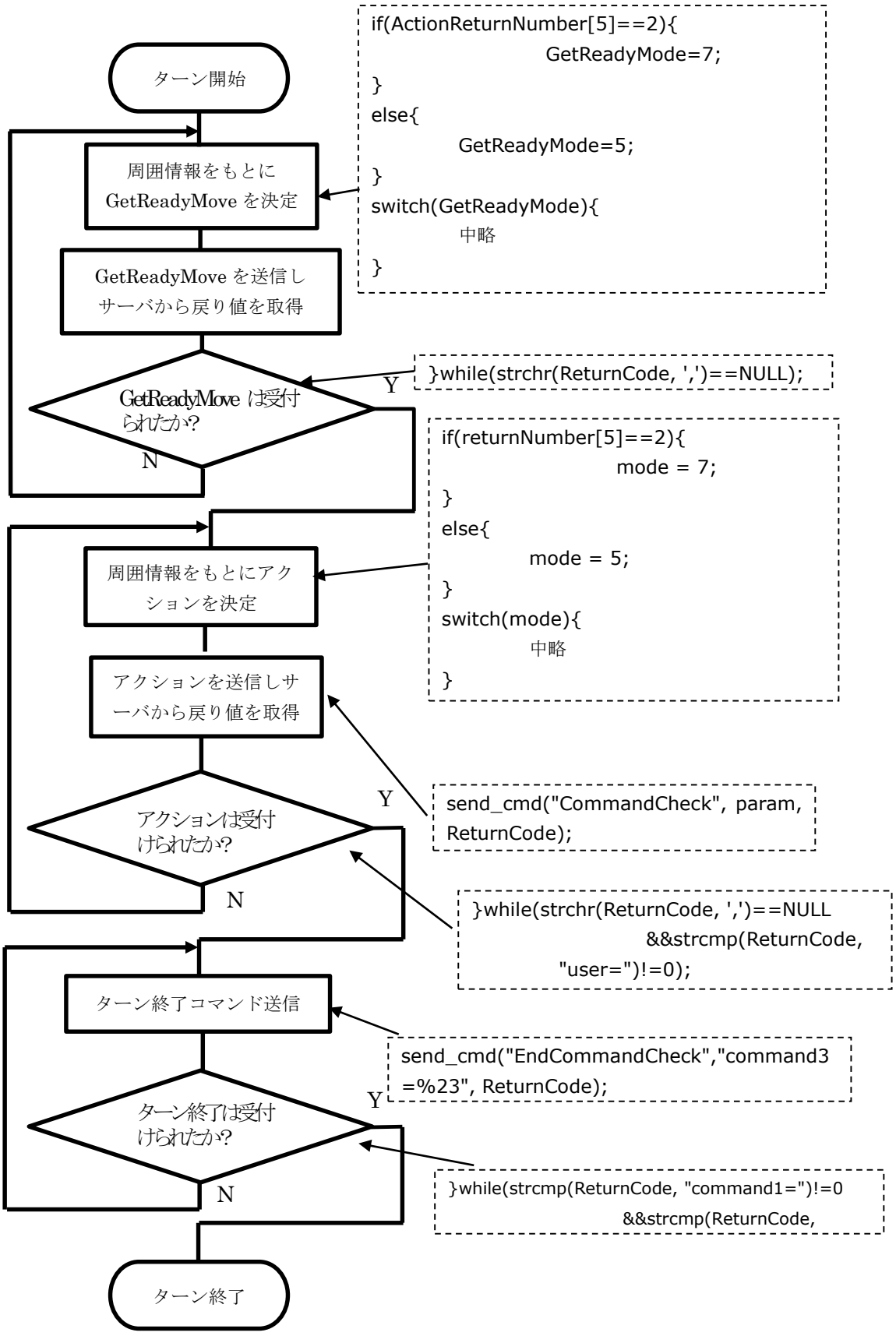
C　終了コマンド（自分のターンを終了させる）	
コマンド名	機能
#	自分のターンを終了させる ※周囲情報なし

10 プログラムの流れについて

みなさんが作るクライアントプログラムと競技サーバのやりとりは次のようになります。まず WEB の仕組みとしてサーバへの接続、WEB コマンドの送信、サーバからのリターンメッセージの受信とサーバからの切断がひとつのセットとなります。複数のセットを連携させる仕組みとして「セッション」を利用します。

準備コマンドの GetReadyMove をサーバへ送信して周囲情報を取得するセット､動作コマンドを選んで送信して周囲情報を取得するセット、ターンを終了させるセットの３つのセットをひとつのパッケージとします。このパッケージをターン終了まで繰り返します。クライアントプログラムは図 1 のフローチャートのようになります。

みなさんは、サーバから得た周囲情報をもとにコマンドを選んで送信する部分を考えます。

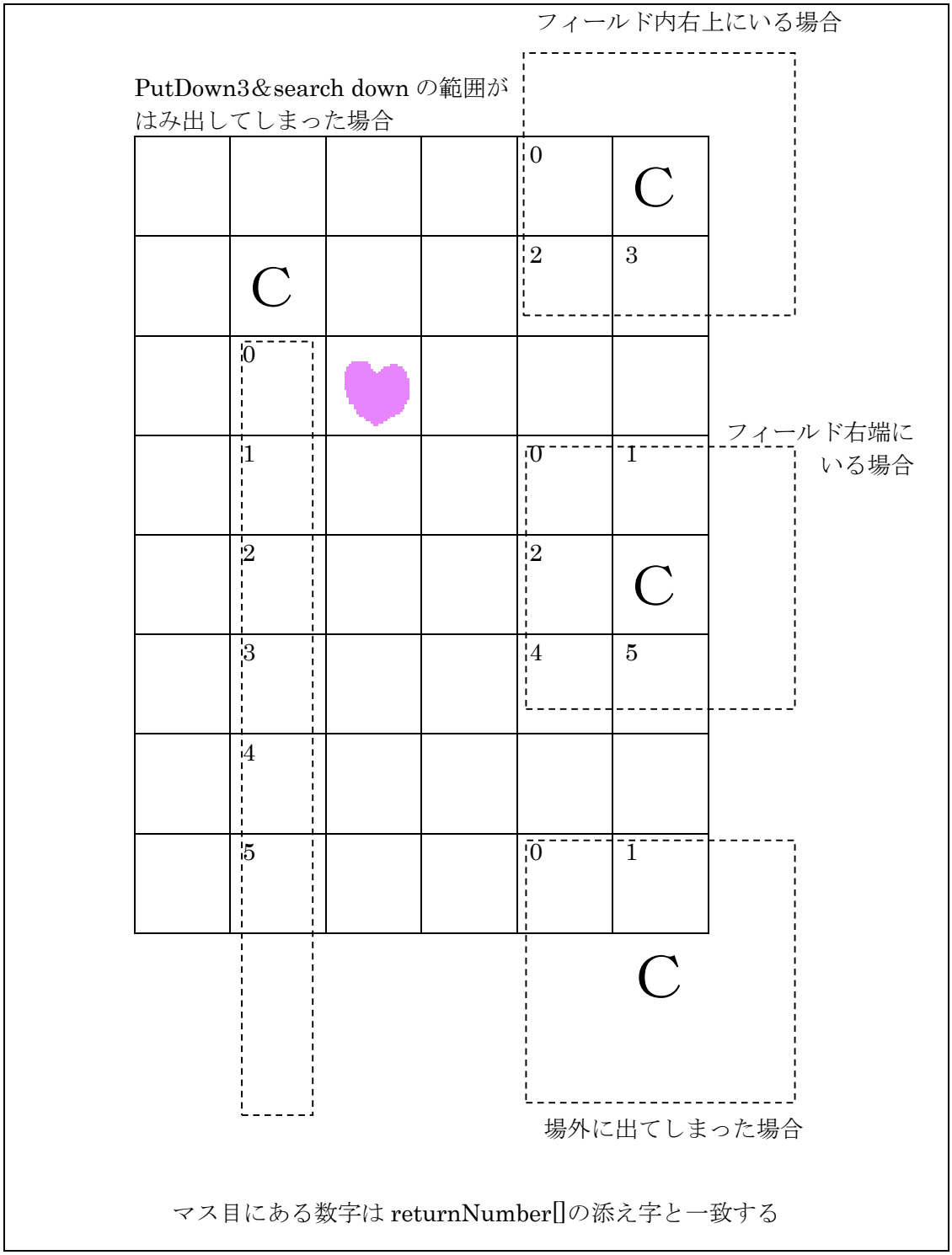


【図 1】プログラムの大まかな流れ

11 サンプルプログラム２（フィールド端を捉える）

①場外について

フィールド外に近づいた場合の例



① プログラムの保存
　サンプルプログラム２をダウンロードするか、アクション部分を打ち直します。
プログラム名は「CHaserOnlineClient2018public002.c」です。

② プログラムの動作
・まず GetReadyMoveRight と WalkRight を使って移動します。
・右端までたどり着いたら下へ移動します。
・以降は角まで行き右回りに移動します。
・mode(モード)、GetReadyMode、count、ActionCount、CountBuff という変数を作り、動作の種類を記憶させています。

③ 対戦(１台のパソコンで二つのクライアントを起動する場合)
・端末を二つ起動させる。
・二つの画面でそれぞれのコマンドを入力し、対戦させる。
（ひとつは自分の ID で起動させ、もうひとつは cool や hot などの公開されている ID で起動する。）

起動コマンドの例(自分の ID)

```
./CHaserOnlineClient2018public002.o␣http://www7019ug.sakura.ne.jp:80/
CHaserOnline003/user/␣-u␣Jibun␣-p␣Watashi␣-r␣1050␣-x␣192.168.30.251:8080
```

起動コマンドの例(もうひとつの ID)

```
./CHaserOnlineClient2018public002.o␣http://www7019ug.sakura.ne.jp:80/
CHaserOnline003/user/␣-u␣cool␣-p␣cool␣-r␣1050␣-x␣192.168.30.251:8080
```

※ 先に接続したクライアントから順に C、H のキャラクタが割振られます。また、戻り値はそれぞれ 1000、2000 となります。

※ ルームは「待ち合わせ」ページにて有効な番号を選んでください。

④ サンプルプログラム（変数宣言部分）

```
int main(int argc, char *argv[]){

        int       i;
        int       RoomNumber = -1;
        char command[20];
        char param[BUF_LEN];
        char buff[10];
        char ProxyAddress[256];
        int   ProxyPort;
        char UserName[20];
        char PassWord[20];
        char ReturnCode[BUF_LEN];
        int   returnNumber[10];
        int   ActionReturnNumber[10];
        char *pivo;
        int   count = 9;
        int   ActionCount = 9;
        int   CountBuff = 9;
        int   mode=5;
        int   GetReadyMode=5;

        strcpy(ProxyAddress, "");        //初期化
        ActionReturnNumber[0]=-10000;
```

各種動作を行うための変数です。

⑤ プログラム解説（変数宣言部分） 1

```
int main(int argc, char *argv[]){
・
・
・
・
        int   count = 9;
        int   ActionCount = 9;
        int   CountBuff = 9;
```

戻り値の個数の初期値を９にします。

⑥ プログラム解説（変数宣言部分）２

```
int main(int argc, char *argv[]){
・
・
        int   mode=5;
        int   GetReadyMode=5;
```

GetReadyMove、アクションの初期動作を右移動に設定する。

⑦ サンプルプログラム（GetReady を発行する）

```
do{
        printf("¥n¥n¥ndeb191 GetReady¥n");  //デバッグ用この行を削除すると
                                                セグメントエラーになる

        strcpy(param, "command1=");
        if(ActionCount != CountBuff){
        if(ActionCount <= CountBuff){
                switch(mode){
                        case 5:
                                GetReadyMode = 7;
                                mode = 7;
                                break;

                                case 7:
                                GetReadyMode = 3;
                                mode = 3;
                                break;

                                case 3:
                                GetReadyMode = 1;
                                mode = 1;
                                break;

                                case 1:
                                GetReadyMode = 5;
                                mode = 5;
                                break;

                                default:
                                GetReadyMode = 7;
                                mode = 7;
                                break;
                                        }
                                }
                                CountBuff = ActionCount;
```

```
                }


                switch(GetReadyMode){
                        case 0:
                                strcat(param, "gr");
                                break;

                        case 1:
                                strcat(param, "gru");
                                break;

                        case 3:
                                strcat(param, "grl");
                                break;

                        case 5:
                                strcat(param, "grr");
                                break;

                        case 7:
                                strcat(param, "grd");
                                break;

                        default:
                                strcat(param, "gr");
                }
                .
                .
                .
                .
```

⑧　プログラム解説（GetReady を発行する）

```
do{
                strcpy(param, "command1=");
                if(ActionCount != CountBuff){
                        if(ActionCount <= CountBuff){
                                .
                                .
                                .
```

戻り値が変化することと曲がるタイミングを判定しています。

⑨　プログラム解説２（GetReady を発行する）

```
        switch(mode){
                        case 5:
                                GetReadyMode = 7;
                                mode = 7;
                                break;

                        case 7:
                                GetReadyMode = 3;
                                mode = 3;
                                break;

                        case 3:
                                GetReadyMode = 1;
                                mode = 1;
                                break;

                        case 1:
                                GetReadyMode = 5;
                                mode = 5;
                                break;

                        default:
                                GetReadyMode = 7;
                                mode = 7;
                                break;
                }
```

右回りに変換しています。

⑩　プログラム解説３（GetReady を発行する）

```
・
・
・
}
                        CountBuff = ActionCount;
                }
```

戻り値の個数を保存します。

⑪　プログラム解説４（GetReady を発行する）

```
switch(GetReadyMode){
                                case 0:
                                        strcat(param, "gr");
                                        break;

                                case 1:
                                        strcat(param, "gru");
                                        break;

                                case 3:
                                        strcat(param, "grl");
                                        break;

                                case 5:
                                        strcat(param, "grr");
                                        break;

                                case 7:
                                        strcat(param, "grd");
                                        break;

                                default:
                                        strcat(param, "gr");
```

GetReadyMode を GetReadyMove に変換します。

⑫　サンプルプログラム（Action を発行する）

```
/*----------------------
    Action を発行する
-----------------------*/
do{
                        strcpy(param, "command2=");
                        if(count != CountBuff){
                                if(count <= CountBuff){
                                        switch(GetReadyMode){
                                                case 5:
                                                        GetReadyMode = 7;
                                                        mode = 7;
                                                        break;

                                                case 7:
                                                        GetReadyMode = 3;
                                                        mode = 3;
                                                        break;

                                                case 3:
                                                        GetReadyMode = 1;
                                                        mode = 1;
                                                        break;

                                                case 1:
                                                        GetReadyMode = 5;
                                                        mode = 5;
                                                        break;

                                                default:
                                                        GetReadyMode = 7;
                                                        mode = 7;
                                                        break;
                                        }
                                }
                                CountBuff = count;
                        }

                        switch(mode){
                                case 1:
                                        strcat(param, "wu");
                                        break;

                                case 3:
                                        strcat(param, "wl");
```

```
                                        break;

                                case 5:
                                        strcat(param, "wr");
                                        break;

                                case 7:
                                        strcat(param, "wd");
                                        break;

                                default:
                                        strcat(param, "wr");
                        }
                        send_cmd("CommandCheck", param, ReturnCode);
                }while(strchr(ReturnCode,       ',')==NULL&&strcmp(ReturnCode,
"user=")!=0);   //Action が受け付けられるまでループ
                ActionCount = returnCode2int(ReturnCode, ActionReturnNumber);
```

各種動作をここで書きます。

⑬　プログラム説明（Action を発行する）

```
do{
                        strcpy(param, "command2=");
                        if(count != CountBuff){
                                if(count <= CountBuff){
        .
        .
        .
}
```

戻り値が変化することを判定し、フィールド内からフィールドの端また横から角への変化を判定するようにしています。

⑭　プログラム説明２（Action を発行する）

```
switch(GetReadyMode){
                                        case 5:
                                                GetReadyMode = 7;
                                                mode = 7;
                                                break;

                                        case 7:
                                                GetReadyMode = 3;
                                                mode = 3;
                                                break;

                                        case 3:
                                                GetReadyMode = 1;
                                                mode = 1;
                                                break;

                                        case 1:
                                                GetReadyMode = 5;
                                                mode = 5;
                                                break;

                                        default:
                                                GetReadyMode = 7;
                                                mode = 7;
                                                break;
                                }
```

右に移動中なら下移動に切り替えます。下に移動中なら左へ、左に移動中なら上へという動きになります。つまり右回りに移動していく事となります。

⑮　プログラム説明３（Action を発行する）

```
}
                                CountBuff = count;
                        }
```

戻り値の個数を保存

⑯　プログラム説明４（Action を発行する）

```
switch(mode){
                        case 1:
                                strcat(param, "wu");
                                break;

                        case 3:
                                strcat(param, "wl");
                                break;

                        case 5:
                                strcat(param, "wr");
                                break;

                        case 7:
                                strcat(param, "wd");
                                break;

                        default:
                                strcat(param, "wr");
                }
```

mode をアクションコマンドに変換します。

⑰　プログラム説明５（Action を発行する）

```
ActionCount = returnCode2int(ReturnCode, ActionReturnNumber);
```

アクションの戻り値を取り出し個数を代入する

⑱　注意

　このクライアントはターン終了までフィールドを右回りに進みます。アイテム類を判定しませんので、得点は増えません。各自で判断を追加してみてください。

12 クライアントの動作について

各種コマンド等について、使い方のヒントや注意点について説明します。

① GetReadyMove について
このコマンドは準備段階から移動することができるコマンドです。前ターンの戻り値を利用して移動できるので倍速で移動できます。

② Put3&Look および Put3&Serch について
このコマンドはアイテムをおとりにして、離れた場所を探査できます。

③ ワープについて
このワープはコマンドではなくアイテム類であることに注意してください。
ステップアップヒント１の８ページでも触れましたが、このワープを取ろうとすると上下左右１０マス分または５マス分移動することができます。左右への移動で考えると walk や put&walk の１０ターン分を１ターンで移動することができます。
違うエリアに移動したいときに使うと良いでしょう。ただし、移動先に他クライアントがいる可能性も考えられますので注意が必要となります。

④ put&walk 系コマンドについて
このコマンドは１ターンで put と walk の二つの動作を組み合わせて実行できる効率のよいコマンドです。CHaserOnline では相手に Put をしてもゲームは終了せずにターン終了まで進みます。相手に Put したらその逆に動くことができるこのコマンドは活用の場が多くありそうです。また、上下左右の動きに加えて斜め方向にも動くことができますので walk のみでの移動よりも早く動くことができます。
しかし必ず Put してから、その逆方向に動きますのでアイテムをつぶしてしまう危険性もあります。

⑤ kei 系コマンドについて
このコマンドは walk の３ターン分を１ターンで移動することができます。うまく活用すれば少ないターン数で広範囲の移動や探査が可能になります。ホワイトに囲まれてしまい身動きがとれないときにも効果を発揮するでしょう。しかし移動先は GetReadyMove では見えない場所ですので注意が必要です。

⑥ ブラックスターについて
ブラックスターもコマンドではなくアイテム類となっています。
ブラックスターはワープと違いお互いとの距離が固定されています。どのように配置されているかがわかれば、マップの形状を知る大きな手掛かりになるでしょう。