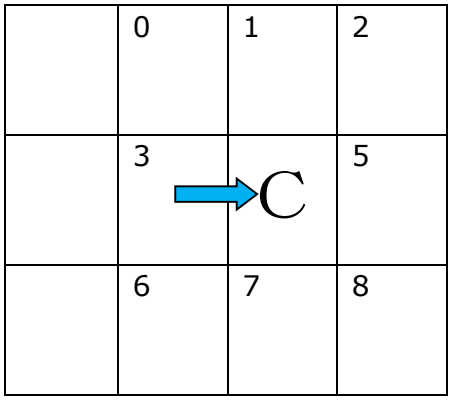
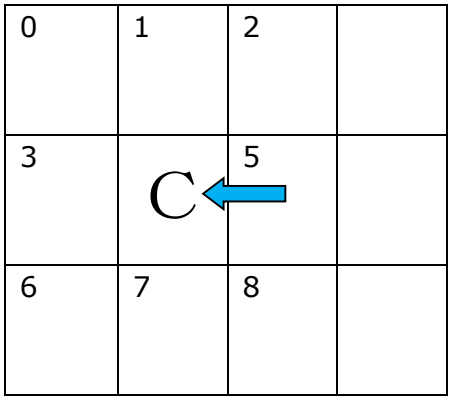
全国高校生プログラミングコンテスト
CHaserOnline
ステップアップヒント２

8 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（2016.6.1 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。

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（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</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。
<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。</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 

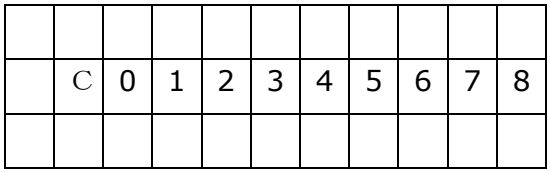
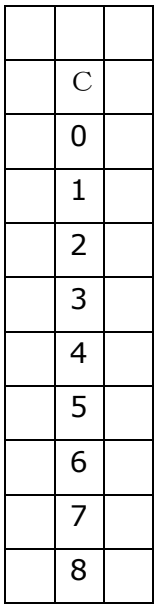
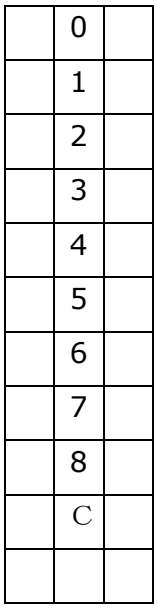
<table>
<tr><td colspan="2">B　動作コマンド look 系（指定した方向の周囲９マスの情報を得る）</td></tr>
<tr><td>コマンド名</td><td>機能</td></tr>
<tr><td>lu
ld
ll
lr</td><td>指定した方向の周囲９マスの情報を得る。

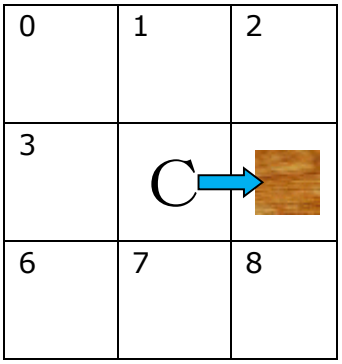
lu・・・上
ld・・・下
ll・・・左
lr・・・右

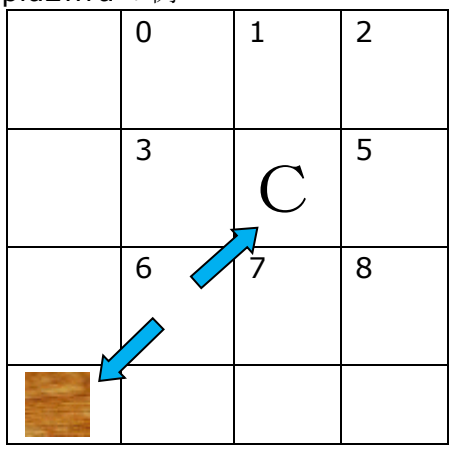
※見えない位置を広く見るので、だいぶ疲労します。

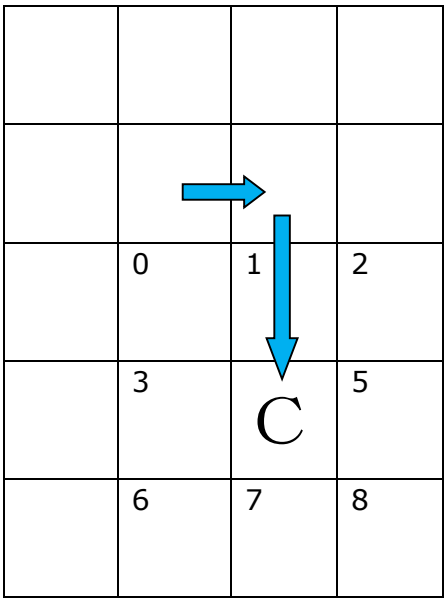
lr の例
<table><tr><td></td><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td></td><td>C</td><td>3</td><td>4</td><td>5</td></tr><tr><td></td><td></td><td>6</td><td>7</td><td>8</td></tr></table>
マス目にある数字は returnNumber[]の添え字と一致する。
また、数字の並びは lu, ld, ll の場合も探査範囲の左上が 0 となる。

ld の例
<table><tr><td></td><td></td><td></td></tr><tr><td></td><td>C</td><td></td></tr><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table></td></tr>
</table>

B　動作コマンド search 系（指定した方向、真っすぐ 9 マスの情報を得る）	
コマンド名	機能
su sd sl sr	指定した方向、真っすぐ 9 マスの情報を得る。 su・・・上 sd・・・下 sl・・・左 sr・・・右 ※見えない位置を細く長く見るので疲労の度合いは少ないです。 sr の例  マス目にある数字は returnNumber[]の添え字と一致する。 また、数字の並びは左上のマスが 0 となる。 sd の例  su の例 

B　動作コマンド put2 系（指定した方向へ土を置く）	
コマンド名	機能
pu2 pd2 pl2 pr2	指定した方向へ土を置く。 pu2・・・上 pd2・・・下 pl2・・・左 pr2・・・右 ※せっかく掘った所を埋めてしまうので、かなり疲労します。 pr2 の例  ※  は土 マス目にある数字は returnNumber[]の添え字と一致する。

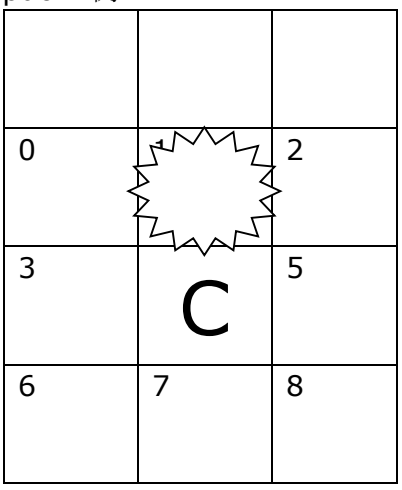
B　動作コマンド put2&walk 系（指定した方向へ土を置き、逆向きに１マス移動する）	
コマンド名	機能
pu2wd pd2wu pl2wr pr2wl pru2wld plu2wrd prd2wlu pld2wru	指定した方向へ土を置き、逆向きに１マス移動する。 pu2wd・・・上に土を置き、下へ移動 pd2wu・・・下に土を置き、上へ移動 pl2wr・・・左に土を置き、右へ移動 pr2wl・・・右に土を置き、左へ移動 pru2wld・・・右上に土を置き、左下へ移動 plu2wrd・・・左上に土を置き、右下へ移動 prd2wlu・・・右下に土を置き、左上へ移動 pld2wru・・・左下に土を置き、右上へ移動 ※土を置きなおかつ一歩下がりますが、ｐｕｔ２よりも疲労しません。また、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。

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。 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。

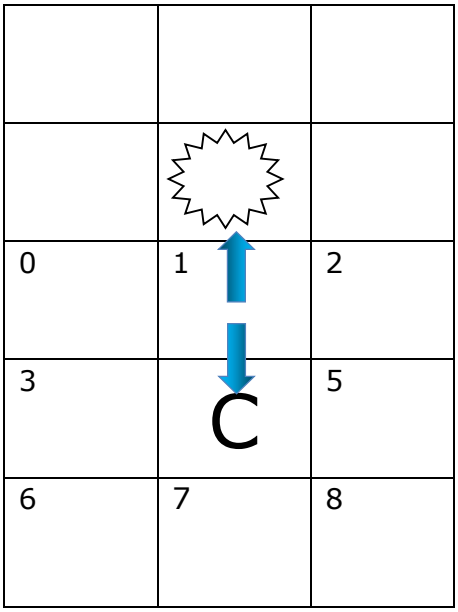
<table>
<tr><td colspan="2">B　動作コマンド put0 系（砕く動作）</td></tr>
<tr><td>コマンド名</td><td>機能</td></tr>
<tr><td>pr0
pl0
pu0
pd0</td><td>指定した方向の土を砕きます。

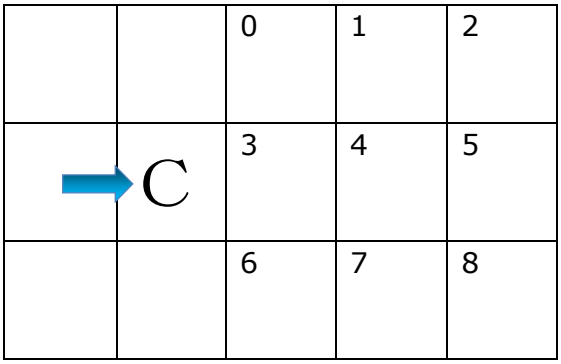
pr0・・・右にある土を砕きます。
pl0・・・左にある土を砕きます。
pu0・・・上にある土を砕きます。
pd0・・・下にある土を砕きます。

※ 砕く動作ですので、疲労がたまりま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（指定した方向の土を砕き、逆向き１マス移動する）	
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逆向きに１マス移動する。 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下に移動する。 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。

B　動作コマンド dig 系（1 歩動いて指定した方向の周囲９マスの情報を得る）	
コマンド名	機能
du dd dl dr	１歩動き、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。 ※一歩動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  マス目にある数字は returnNumber[]の添え字と一致する。

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

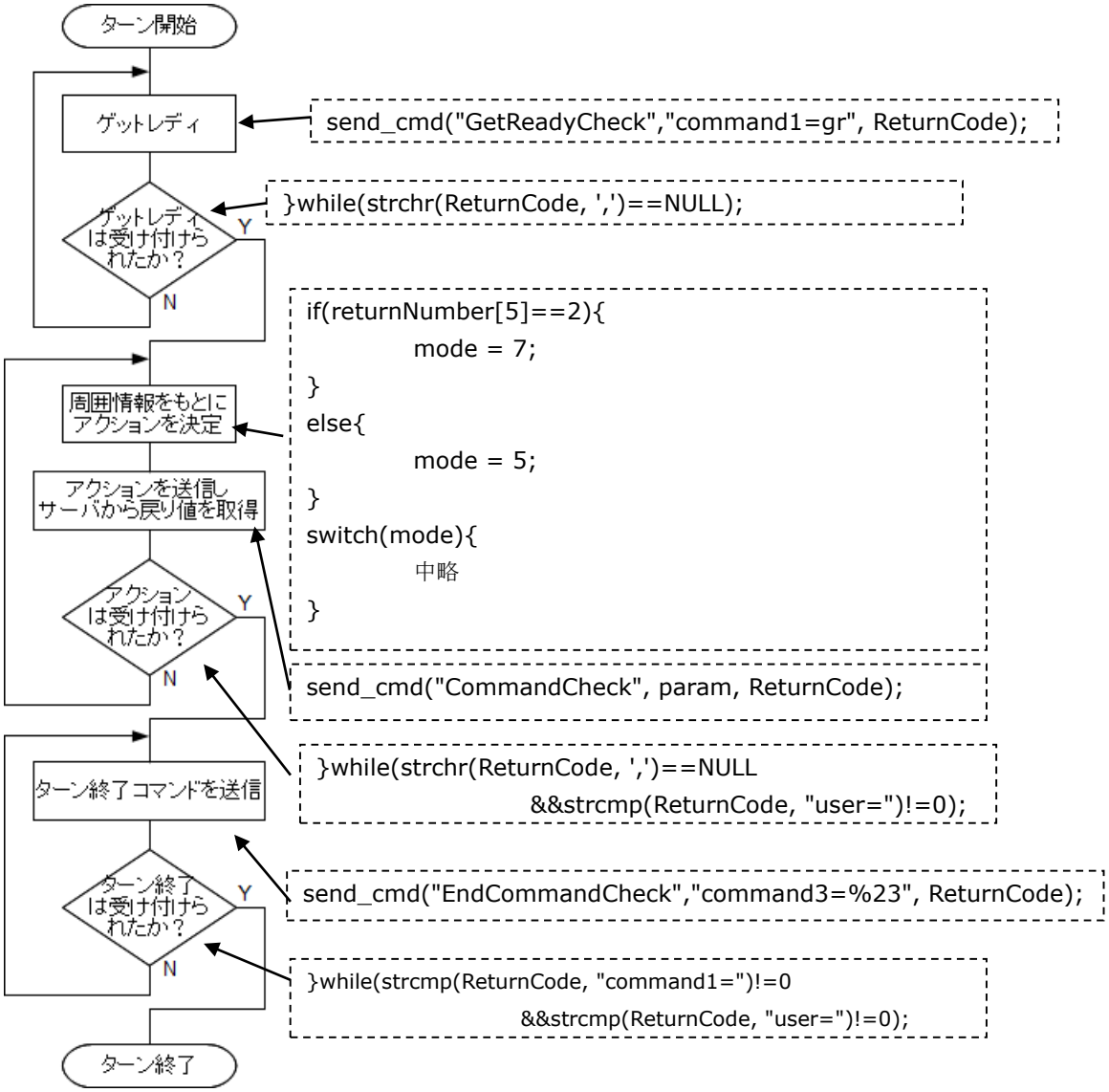
9 プログラムの流れについて

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ｇｒ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。

【フローチャート】　　【プログラム】(途中省略あり)



【図 1】プログラムの大まかな流れ

10 サンプルプログラム２（ｉｆ文の使い方）

① プログラムの保存
サンプルプログラム２－１～２－８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「CHaserOnlineClient2015public002-1.c」から
「CHaserOnlineClient2015public002-8.c」です。

② プログラムの動作
１～５は、隣のマスが土だったら進み、６～８は if と else の組み合わせ方についてのサンプルです。

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)

```
./CHaserOnlineClient2016public002-1.o␣http://www7019ug.sakura.ne.jp:80/
CHaserOnline003/user/␣-u␣Jibun␣-p␣Watashi␣-r␣141␣-x␣192.168.30.251:8080
```

自分のＩＤ　自分のパスワード　ルーム番号　プロキシサーバアドレス　プロキシポート番号

起動コマンドの例(もうひとつの ID)

```
./CHaserOnlineClient2016public002-1.o␣http://www7019ug.sakura.ne.jp:80/
CHaserOnline003/user/␣-u␣cool␣-p␣cool␣-r␣141␣-x␣192.168.30.251:8080
```

公開用 ID の cool の使用例
※hot を使う場合は差し替えてください

※ 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。

④　サンプルプログラム（「Action を発行する」部分抜粋）
　　次のプログラムに以下 002-1～002-5 のｉｆ文がそれぞれ入ります。

```
/*----------------------
Action を発行する
------------------------*/
do{
        strcpy(param, "command2=");
                .
                .
                .
        switch(mode){
                case 1:
                        strcat(param, "du");
                        break;
                case 3:
                        strcat(param, "dl");
                        break;
                case 5:
                        strcat(param, "dr");
                        break;
                case 7:
                        strcat(param, "dd");
                        break;
                case 10:
                        strcat(param, "keilu");
                        break;
                case 18:
                        strcat(param, "keird");
                        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
```

002-1

```
        if(returnNumber[7]==2){                 //もしも下が土だったら
                mode = 7;                       //下へ
        }
        else{
                mode = 3;                       //左へ
        }
```

002-2

```
if(returnNumber[3]==2){                    //もしも左が土だったら
        mode = 3;                          //左へ
}
else{
        mode = 1;                          //上へ
}
```

002-3

```
if(returnNumber[1]==2){                    //もしも上が土だったら
        mode = 1;                                  //上へ
}
else{
        mode = 5;                                          //右へ
}
```

002-4

```
if(returnNumber[5]==2){                    //もしも右が土だったら
        mode = 5;                                  //右へ
}
else{
        mode = 18;                                         //右下へ
}
```

002-5

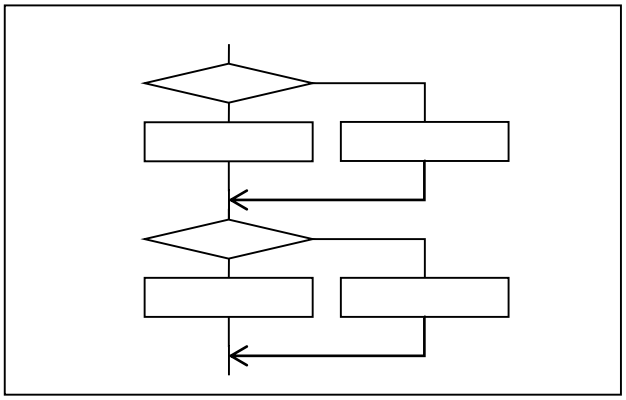
```
if(returnNumber[7]==2){                    //もしも下が土だったら
        mode = 7;                                  //下へ
}
else{
        mode = 10;                                         //左上へ
}
```

⑤　プログラム説明

```
if(returnNumber[○]==2){
        mode = ○;
}
else{
        mode = △;
}
```

002-1～002-3 はすべて、進もうと考えている方向が土だったらその方向へ進み、そうでなければ９０度右回転した方向に進むようにしています。
また、002-4～002-5 は進もうと考えている方向が土だったらその方向へ進み、そうでなければ kei で斜め方向に進むようにしています。

これらのサンプルを単純につなげると以下のフローチャートのようになります。



ｉｆ分の使い方でクライアントの「動き」が変わります。

次にサンプル 002-6～002-8 について説明します。

002-6

```
if(returnNumber[7]==2){                 //もしも下が土だったら
        mode = 7;                               //下へ
}
if(returnNumber[3]==2){                 //もしも左が土だったら
        mode = 3;                               //左へ
}
else{
        mode = 12;                                      //右上へ
}
```

002-7

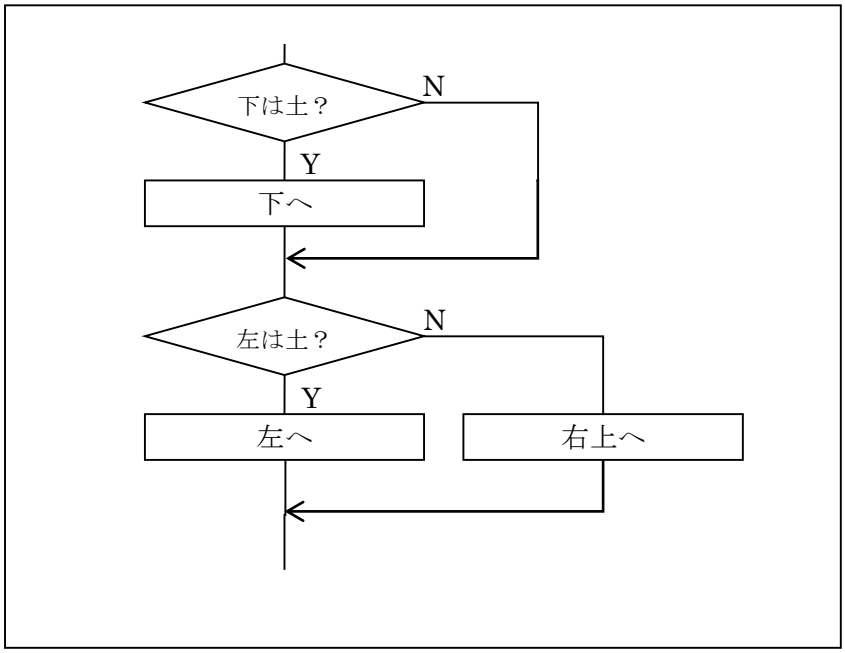
```
if(returnNumber[7]==2){                 //もしも下が土だったら
        mode = 7;                               //下へ
}
else{
        mode = 12;                                      //右上へ
}
if(returnNumber[3]==2){                 //もしも左が土だったら
        mode = 3;                               //左へ
}
```

002-8

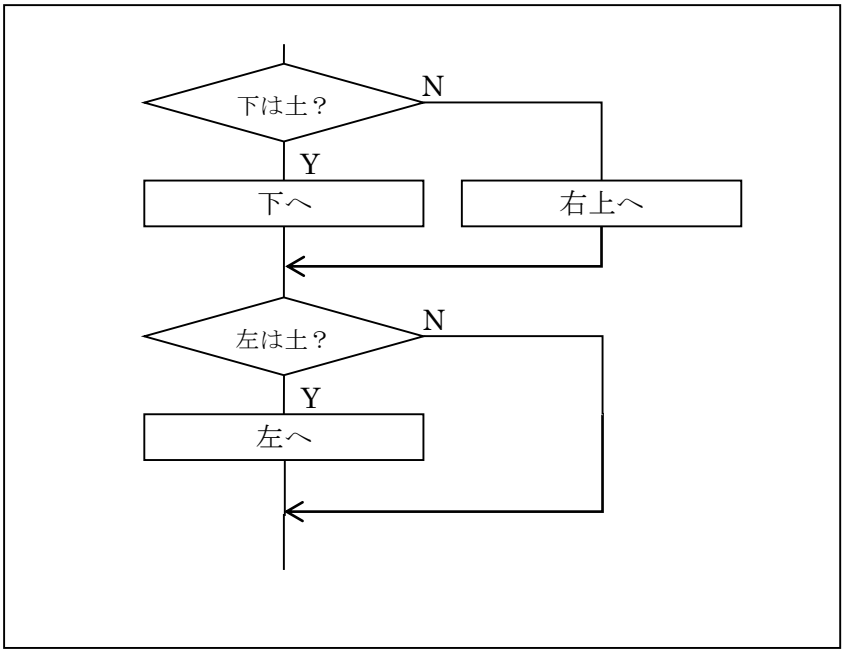
```
if(returnNumber[7]==2){                 //もしも下が土だったら
        mode = 7;                               //下へ
}
else if(returnNumber[3]==2){            //もしも左が土だったら
        mode = 3;                               //左へ
}
else{
        mode = 12;                                      //右上へ
}
```

コードだけで見ると行の位置が変わっているだけでほとんど同じように見えます。
ここでフローチャートで考えてみると以下のようになります。

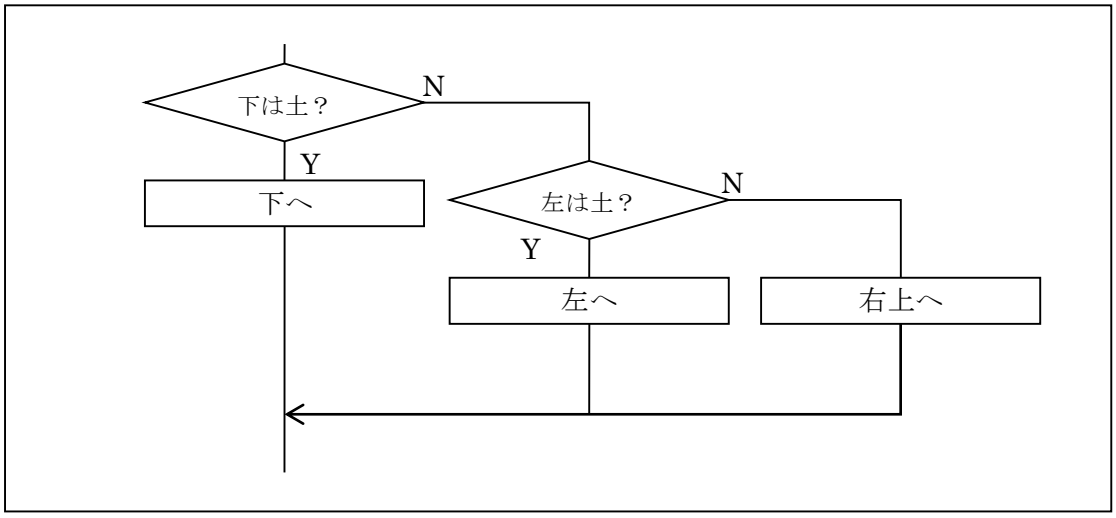
002-6



002-7



002-8



⑥動作結果の違い
サンプル 002-6～002-8 において次のようなマップを考えたときのサンプルごとの動作結果は以下のようになります。
マップ 1

002-6　左へ
002-7　左へ
002-8　左へ

マップ 2

002-6　左へ
002-7　左へ
002-8　下へ

マップ 3

002-6　右上へ
002-7　下へ
002-8　下へ

⑥002-6、002-7 は条件が下に書いてあるほど優先され、002-8 は上に書いてあるほど優先されることが分かります。クライアントプログラムを作るときに自分の作戦がどのように決定させようしているかをはっきりさせておかないと、思わぬ動きをしてしまいます。
また、化石やアイテムなどについても判定に加え、より高得点が取れるクライアントにしましょう。

11 クライアントの動作について

ワープや put&walk 系コマンド、kei 系コマンド、dig 系コマンドについて、使い方のヒントや注意点について説明します。

① 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。
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。

② 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 Put2 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。

しかし注意しなくてはならないのは今年から put0 系が加わりました。put0 系コマンドは土を砕くことができる有用な行動となりますが、put2 系はせっかく掘った場所でも土をかぶせてしまうので良くない行動として分類させるようになりました。

そして必ず 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。

③ 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。しかし移動先はゲットレディでは見えない場所ですので注意が必要です。

④ dig 系コマンドについて

このコマンドは look と walk の動作を組み合わせて実行できるコマンドです。look では留まっている場所から 3×3 マスしか探査することはできませんが、dig を使うと一歩進んで探査することができるようになります。実行することで look よりさらに先を探査できるようになり、相手より先にアイテム等を探せるチャンスをつかめるかもしれません。

⑤ 参考ページについて

過去のステップアップヒントも引き続き公開しています。
是非参考にしてください。

場外に出ない方法	マップの作り方
ローカルサーバの使い方	シェルスクリプトについて、自分の位置を検知する