全国高校生プログラミングコンテスト

CHaserOnline 2022

STEP UP HINT … [2]

今回のヒントは、「プログラムを動かしてみる」から、「プログラムをレベルアップさせる」ための知識や内容についての解説です。
少しでも自分流にプログラムを変えるだけで、他の人と差をつけることができるぞ！

目次

８プログラムの流れ（コマンド実行の流れ）
９コマンド一覧（2022.6.22）
１０コマンドの動作・MAP の機能について
１１サンプルプログラム７（フィールド端を捉える）の解説
１２プログラムの改良 part2

８プログラムの流れ（コマンド実行までの流れ）

ヒント１では、主にクライアントプログラムの動かし方を解説しました。そのプログラムと通信をしている競技サーバのやりとりは以下のようになっています。

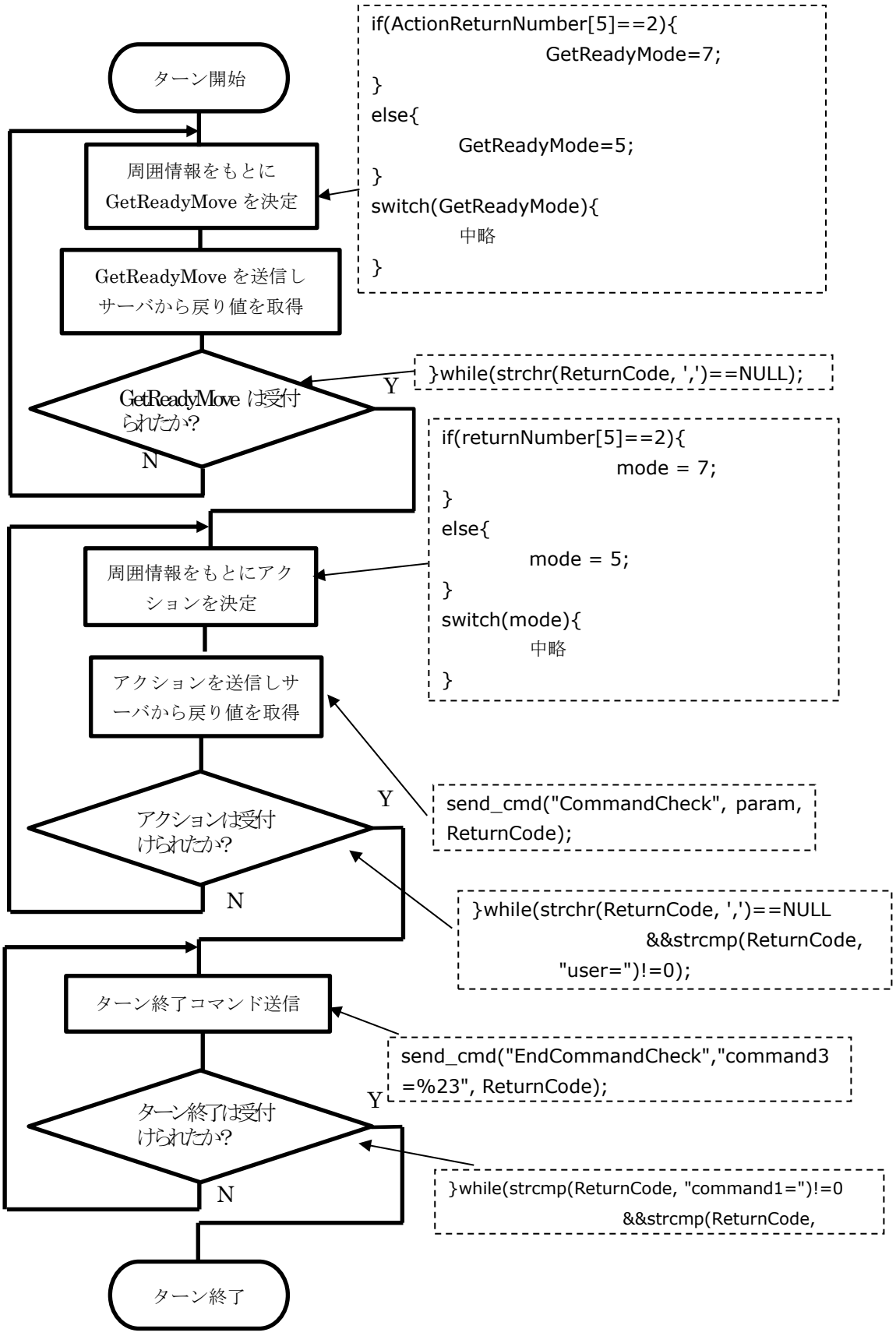
サーバへの接続、コマンドの送信、サーバからのリターンメッセージの受信、サーバからの切断。以上がサーバとの細かいやりとりの流れになります。

Ａ：準備コマンド（GetReadyMove）をサーバへ送信して周囲情報を取得するやりとり。
Ｂ：動作コマンドを選んで送信して周囲情報を取得するやりとり。
Ｃ：終了コマンドを送るターン終了のやりとり。

以上３つのやりとりが一つのセットとなり、このセットを繰り返します。

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。

さて、今回のヒント２では、プログラムがサーバから得た周囲情報をもとに、「どのようなコマンドを実行しようか。」と考える（判断する）部分を作成することになります。



【図 1】プログラムの大まかな流れ

9 コマンド一覧（2022.6.22）

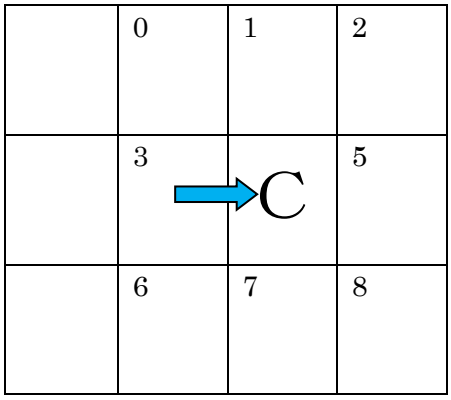
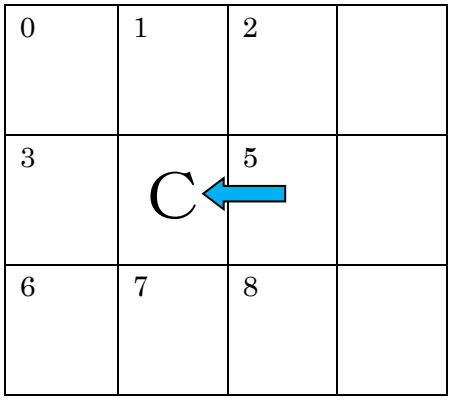
では、具体的にクライアント（プレイヤ）はどのようなコマンドを実行できるのか。一覧を以下に示します。表内の C や H はプレイヤを示しています。

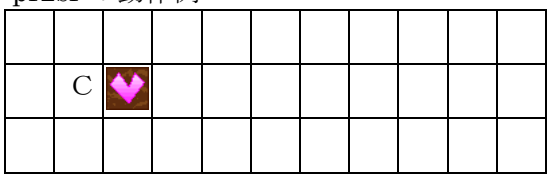
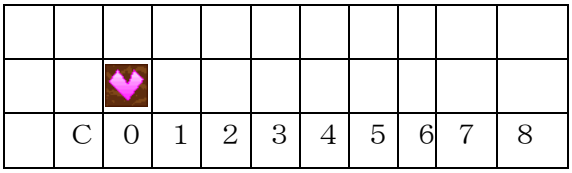
周囲情報を得るコマンドの返り値は、整数型の配列 returnNumber[10]に入ります。
また、各コマンドを実行するたび、コマンド毎に設定された疲労（減点）があります。効率よく得点を稼ぐには、プログラムによく考えさせる必要があります。

※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>

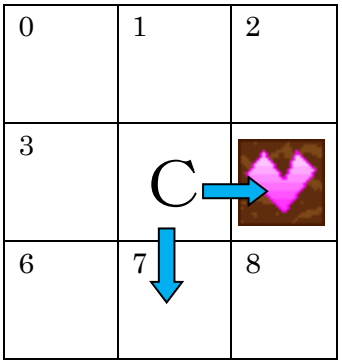
B　動作コマンド walk 系（指定した方向へ 1 マス移動する）	
コマンド名	機能
wu wd wl wr	指定した方向へ 1 マス移動する。 wu・・・上 wd・・・下 wl・・・左 wr・・・右 ※各種のアイテム等を探しに歩くので、アイテムを獲ない限り疲労がたまっていきます。 wr の例  マス目にある数字は returnNumber[]の添え字と一致する。 また、数字の並びは wu, wd, wl の場合も移動先の周囲 9 マスの左上が 0 となる。 wl の例 

B　動作コマンド Put&Search 系 （指定した方向へアイテムを置き、 右方向のマスに移動し、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 ↓指定方向右マスに移動 ↓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。

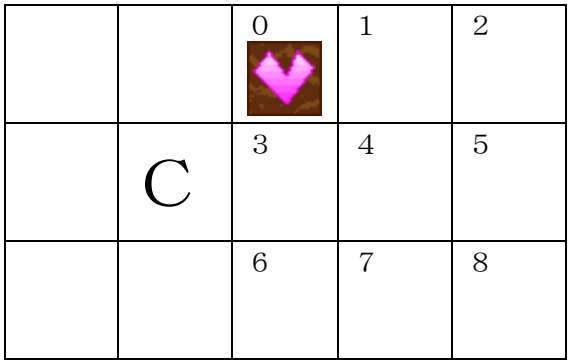
B　動作コマンド Put&Look 系
（指定した方向へアイテムを置き、右方向のマスに移動し周囲情報を得る）

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



↓指定方向右マスに移動
↓周囲９マスの情報を得る

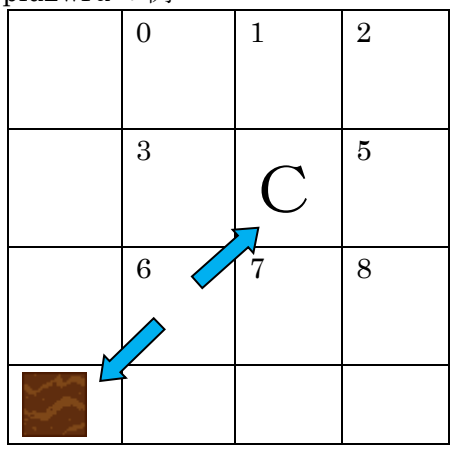


マス目にある数字は returnNumber[]の添え字と一致する。

<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。

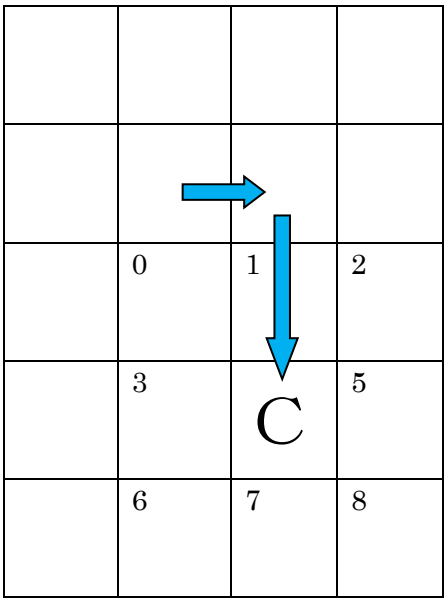
pu2wd・・・上に土を置き、下へ移動
pd2wu・・・下に土を置き、上へ移動
pl2wr・・・左に土を置き、右へ移動
pr2wl・・・右に土を置き、左へ移動
pru2wld・・・右上に土を置き、左下へ移動
plu2wrd・・・左上に土を置き、右下へ移動
prd2wlu・・・右下に土を置き、左上へ移動
pld2wru・・・左下に土を置き、右上へ移動

※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

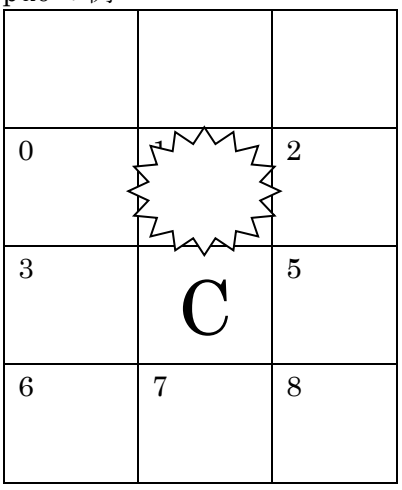
マス目にある数字は returnNumber[]の添え字と一致する。</td></tr>
</table>

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。

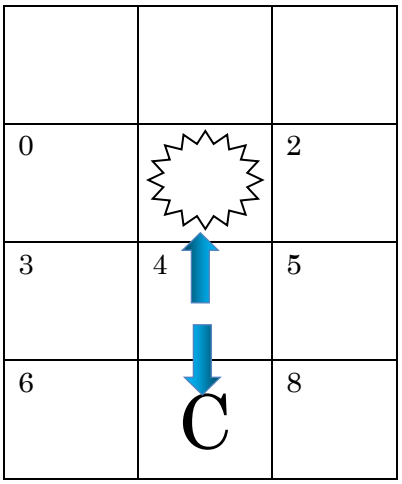
pu0 の例


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

<table>
<tr><td colspan="2">B　動作コマンド系（砕く動作）</td></tr>
<tr><td>コマンド名</td><td>機能</td></tr>
<tr><td>pu0wd
pd0wu
pl0wr
pr0wl
pru0wld
plu0wrd
prd0wlu
pld0wru</td><td>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。

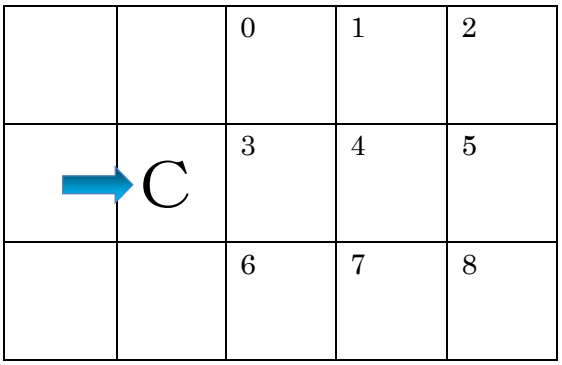
pu0wd の例


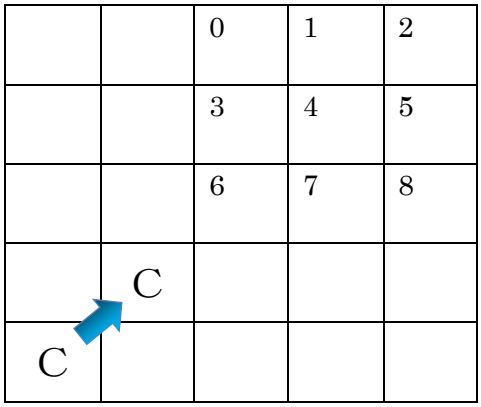
マス目にある数字は returnNumber[]の添え字と一致する。</td></tr>
</table>

<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。

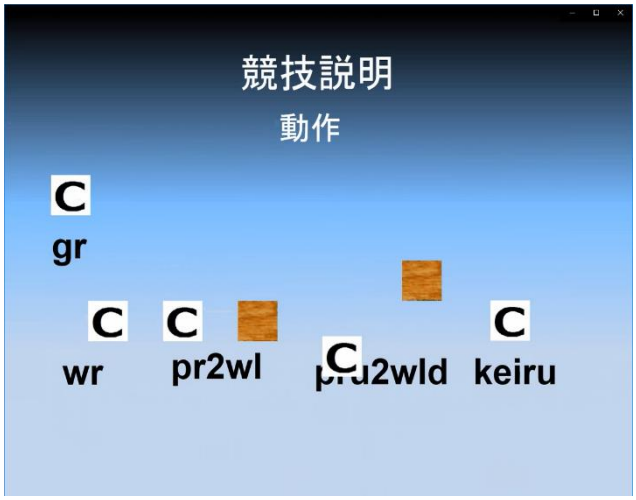
dr の例


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

参考動画



Puppy750 053　全国審判長 ppt03
https://www.youtube.com/watch?v=6CGbNTj8hkc

Puppy750 053　全国審判長 ppt03 long part1
https://www.youtube.com/watch?v=s7kO0BfCojw

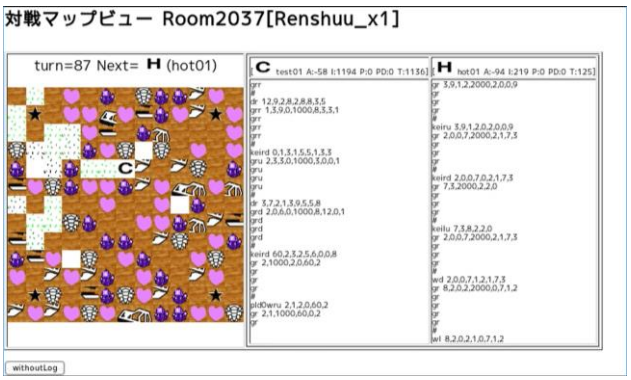
Puppy750 053　全国審判長 ppt03 long part2
https://www.youtube.com/watch?v=5Ss6at2wuco

１０コマンドの動作・MAP の機能について

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 コマンドについて
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。

参考動画（cool が倍速で動きます）



Puppy750 042 TestVs サンプル 14 Room2037 1
https://www.youtube.com/watch?v=i0odNIusFhs&t=86s

② Put3&Look および Put3&Serch コマンドについて
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。

参考動画



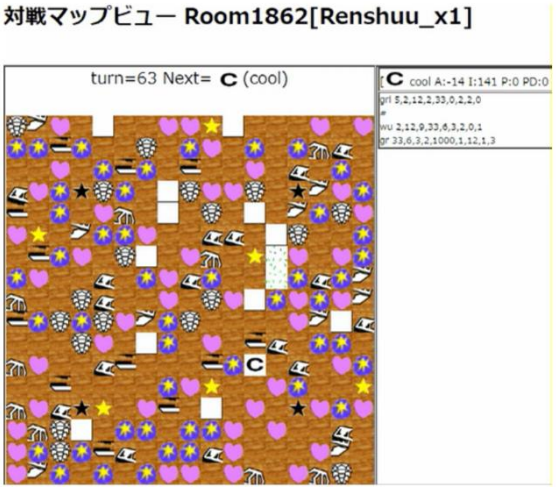
Puppy750 025 サンプル 3 Room1702 1
https://www.youtube.com/watch?v=jHH4eDgVKKo

③　ワープについて

　このワープは、コマンドではなく MAP に配置されているものです。
ステップアップヒント １ にもありますが、このワープマスへ移動すると上下左右１０マス分または５マス分移動することができます。つまり、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。

　ある程度探索を終えた時など、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です。

参考動画



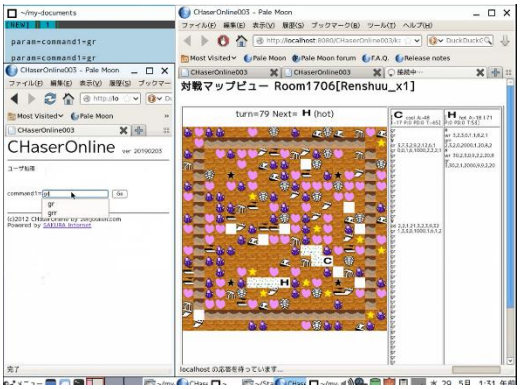
ワープの戻り値は？002 1
https://www.youtube.com/watch?v=6VjHRaUH--0

④　put&walk 系コマンドについて

　これらのコマンドは、1 ターンで put と walk の 2 つの動作を実行できる効率のよいコマンドです。上下左右の動きに加えて斜め方向にも動くことができるため、walk のみでの移動よりも早く動くことができます。

　しかし、必ずアイテムを Put してから逆方向に移動するので、アイテムをつぶしてしまう危険性や、アイテムを相手に取られてしまう可能性もあります。

参考動画



Puppy750_058 手動による Put2andWalk
https://www.youtube.com/watch?v=QYkSRelCxyM

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。

　低ポイントのマス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できます。しかし、移動先は GetReadyMove で見えない場所なので注意が必要です。

参考動画

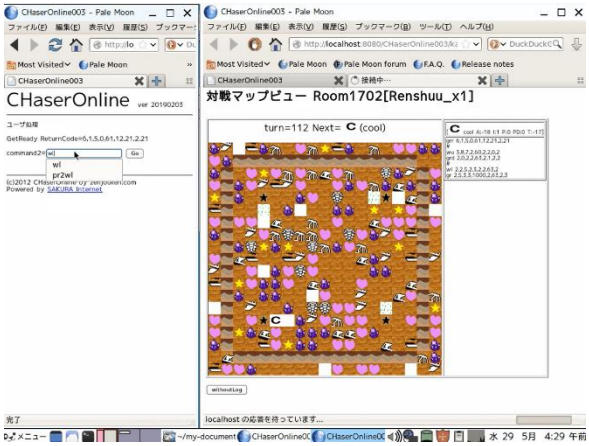


Puppy750 037 サンプル 13 Room1702 1
https://www.youtube.com/watch?v=hlh8EmASQ0Q

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。
ブラックスターもワープのような移動をするマスですが、ワープとは違い移動先の距離が固定されています。どのようにこのマスが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。

参考動画



Puppy750_059 手動によるブラックスター
https://www.youtube.com/watch?v=hsHrMjBbGOA

⑦　今年度からの仕様②

　ステップアップヒント１の方では、ストーリーに沿った新要素の追加や変更について解説して行きました。こちらではさらに、動作に関する変更についてお知らせします。

【ストーリー】（一部抜粋）

今回はジオードが集中するエリアを発掘することになったから、ジオードの殻を砕くためにマシーンの直線掘削力を強化するチューニングしておいた。その一方、他の作業の効率が下がったような…… まぁ、良しとしよう！

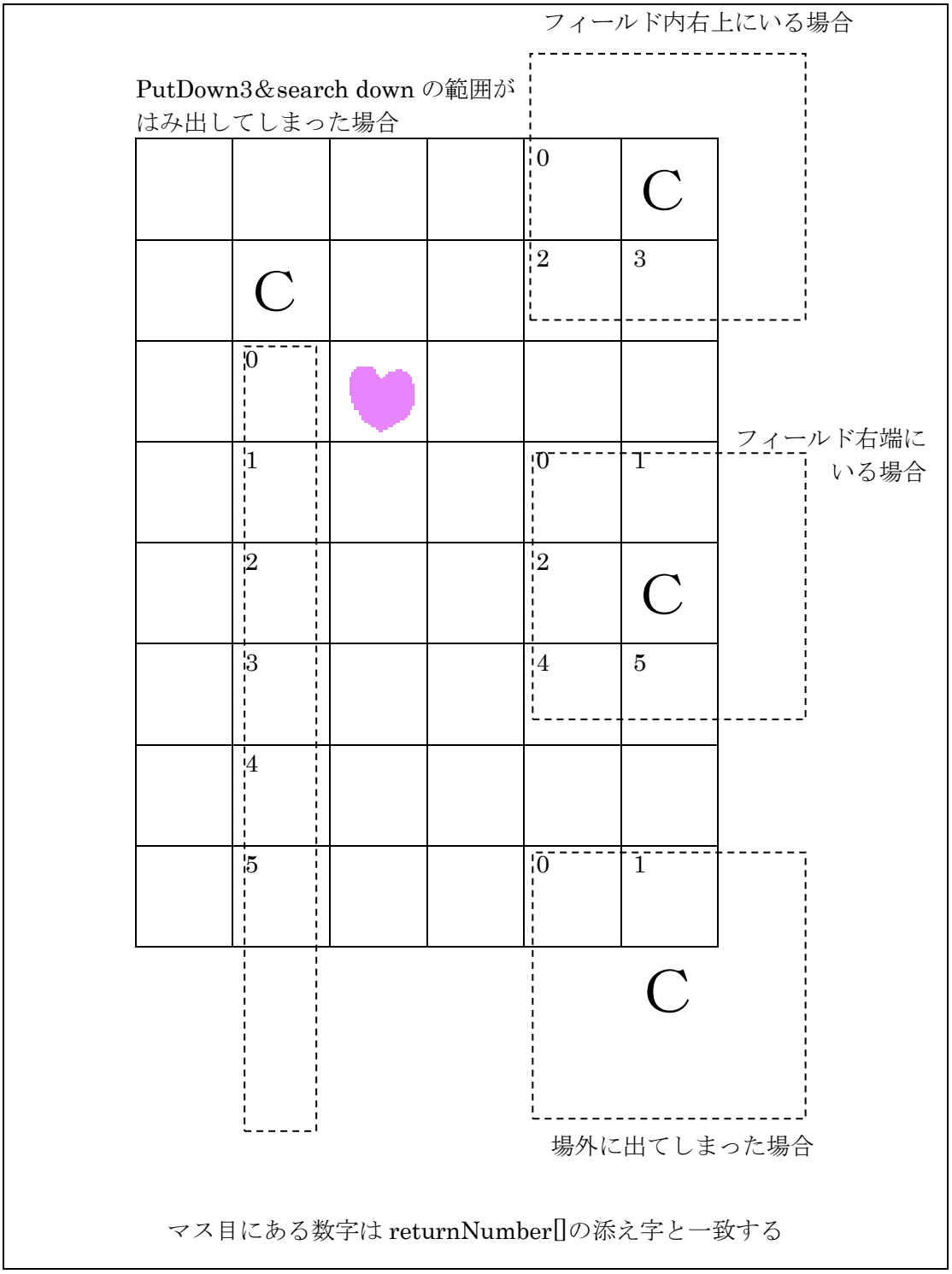
今年度は、直進に特化したチューニングということで、行動による疲労が去年とは変更になっています。直線の移動（上下左右）に関しては強化されていますが、他の作業（コマンド）に関しては効率が落ち、疲労度が大きくなってしまいました。直線以外の移動を使用する際は注意しよう！

１１サンプルプログラム７（フィールド端を捉える）の解説

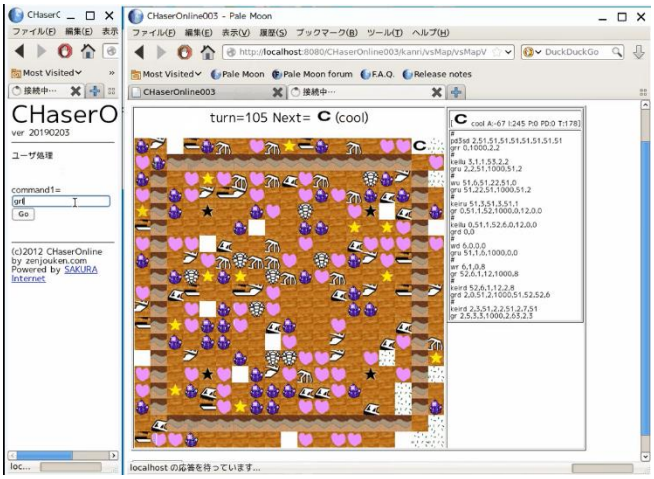
フィールドから落ちないようにするには…？

①場外について

フィールド外に近づいた場合の例



参考動画



Puppy750 056 手動クライアント 3 フィールド端、サーチ
https://www.youtube.com/watch?v=lJCD2_oakJ0

Puppy750 056 手動クライアント 2 フィールド端の戻り値、GRMove、桂馬、サーチ
https://www.youtube.com/watch?v=KzMmNssWe3M

Puppy750 056 手動クライアント 1 接続、桂馬
https://www.youtube.com/watch?v=Sb6wF6m6UmI

★プログラムを動作させる方法（ヒント 1 を参照）と内容の解説
※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ないので、得点は増えません。足りない部分は各自で改良していくことが大事です。

① プログラムの保存
サンプルプログラム 7 をダウンロードするか、自身のプログラムを打ち直します。
サンプルプログラムの名は「CHaserOnlineClient2022public007.c」です。

② サンプルプログラムの動作解説
・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 自分だけの対戦(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（terminal やコマンドプロンプト）を 2 つ起動する。
・二つの画面でそれぞれ対戦するときのコマンドを入力すると対戦が始まる。
（片方は自分の ID で起動し、もう片方は cool や hot 等の練習用 ID で起動する。）

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     Retsu, Gyou;
        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];
        int   HoseiReturnNumber[19][19];
        char *pivo;
        int   count;
        int   ActionCount = 9;
        int   CountBuff = 9;
        int   mode=5;
        int   GetReadyMode=5;
        int   wait_flag = 1;
        int   wait_time = 0;
        int   BreakLimitMax = 0;
        int   BreakLimit = 0;
        int   OutSide=1;
        int   ViewMode = 0;

        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;
        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 4:
                                        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");
}
```

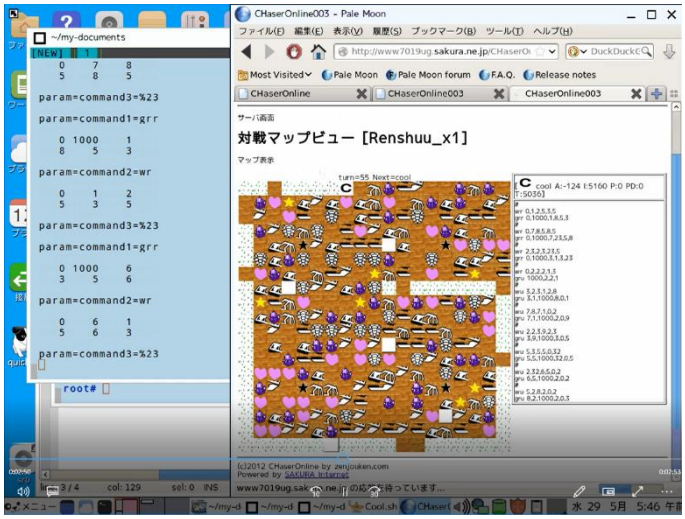
mode をアクションコマンドに変換します。

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

参考動画



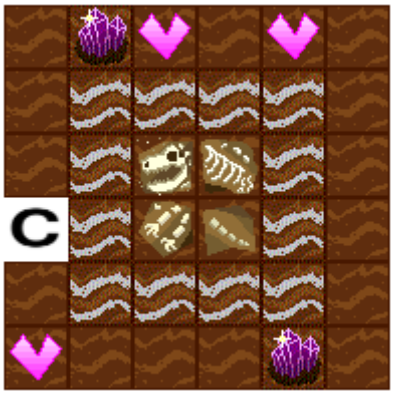
Puppy750_063 サンプルプログラム 2 の動作_ルーム 2416

https://www.youtube.com/watch?v=TH10t_QqX_Q

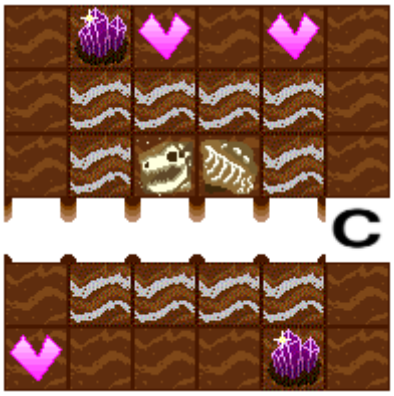
１２プログラムの改良 part2

今回は、今年からの新要素「ジオード」に対応したプログラムの作成について考えていこう！

前回のステップアップヒントでも紹介した通り、ジオードを踏んでしまうと大きなマイナスとなってしまいます。前回作成したプログラムや、今回解説したサンプルを改良してライバルに差をつけよう！



この画像が、実際のマップだと考えたときに、今回解説したサンプルを実行すると右に進み続けることになる。５ターン進み続けると下の画像のようになるぞ！



獲得したアイテムは、「ジオード」、「化石３」、「化石４」、「ジオード」、「土」の５つです。合計得点は何点になるだろうか？前回のステップアップヒントに得点が分かる表があるぞ！計算してみよう！

今回紹介したコマンドの中に、指定した方向の土を砕く「put 系」のコマンドがある。これを追加してジオードを砕いてみよう！



★プログラム記入例

```
if(returnNumber[mode]==4){
    mode = mode + 30;
}

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;
    .
    .
    .
    case 31:
            strcat(param, "pu0");
            break;

    case 33:
            strcat(param, "pl0");
            break;

    case 35:
            strcat(param, "pr0");
            break;

    case 37:
            strcat(param, "pd0");
            break;
    default:
            strcat(param, "wr");
}

mode = mode – 30;
```