BREW & EZアプリ
ゲームプログラミング

○　第１０回　キー入力の処理　○

キー入力情報の取得

BREWには、どのキーが押されているかを取得するAPIはありません。その代わりに、キーが押されたり離されたりすると、イベントとしてイベントハンドラに通知されます。

キーに関するイベントは、キーが押されたときに発生するEVT_KEY_PRESS、キーが離されたときに発生するEVT_KEY_RELEASE、キー処理(発生する条件は機種によって異なります)イベントのEVT_KEYがあります。たとえば、数字の１キーを押し、しばらく押し続けてから離すと、イベントハンドラには以下のイベントが送られます。

```
EVT_KEY_PRESS (wParam = AVK_1)  と  EVT_KEY (wParam = AVK_1, dwParam = 0)
      ↓
EVT_KEY (wParam = AVK_1, dwParam = KB_AUTOREPEAT)
      ↓
EVT_KEY (wParam = AVK_1, dwParam = KB_AUTOREPEAT)
      ↓
   ……
      ↓
EVT_KEY_RELEASE (wParam = AVK_1, dwParam = 0)
```

これらのイベントが発生すると、イベントハンドラのwParamには、AVK_SELECTやAVK_CLRといった、どのキーが押されたかを示すコードが渡されます。それぞれのキーに対してひとつのイベントが発生するので、イベントをそのまま利用しても、同時押しを判別することはできません。

ゲームプログラムでは、EVT_KEY_PRESSイベント(またはEVT_KEYイベントとdwParam)とEVT_KEY_RELEASEイベントを使用します。同時押しを判別するため、キーイベントの処理ではキーが押された、キーが離されたといったフラグを立てるだけにします。内部処理で各キーのフラグを判別し、キャラクターの移動などの処理に役立てます。

なお、エミュレータでは、キーボードのキーリピート機能により、キーを押しっぱなしにいるにもかかわらず、EVT_KEY_PRESSイベントが何度も発生してしまいます。

課　題

キーの入力によってキャラクターを移動させましょう。

(１)キーのフラグを格納する構造体を宣言します。以下のプログラムを適切な場所に追加しましょう。

```
// キーフラグ構造体
typedef struct _KEY {
    boolean   UP;       // ↑
    boolean   DOWN;     // ↓
    boolean   LEFT;     // ←
    boolean   RIGHT;    // →
    boolean   SELECT;   // セレクト
    boolean   CLR;      // クリア
    boolean   NUM0;     // 0
    boolean   NUM1;     // 1
    boolean   NUM2;     // 2
    boolean   NUM3;     // 3
    boolean   NUM4;     // 4
    boolean   NUM5;     // 5
    boolean   NUM6;     // 6
    boolean   NUM7;     // 7
    boolean   NUM8;     // 8
    boolean   NUM9;     // 9
    boolean   STAR;     // ＊
    boolean   POUND;    // #
} KEY;
```

主要なキーごとにフラグを割り当てます。条件を満たしていればTRUE、そうでなければFALSEを格納します。

(２)キーが押されたフラグ、キーが離されたフラグ、キーが押されているフラグを格納する変数を宣言します。以下の変数をアプレット構造体に追加しましょう。

```
KEY    keyPress;          // キープレス情報(キーが押されたフラグ)
KEY    keyRelease;        // キーリリース情報(キーが離されたフラグ)
KEY    keyDown;           // キーダウン情報(キーが押されているフラグ)
```

(３)KEY構造体のフラグをクリアする関数を作成します。以下のプログラムを適切な場所に追加しましょう。

```
// キーフラグクリア
void プロジェクト名_KeyClear(KEY* pKey)
{
    MEMSET(pKey, 0, sizeof(KEY));
}
```

(４)(３)のプロトタイプを適切な場所に作成しましょう。

(５)アプリケーションが中断から復帰したとき、キーフラグを無効に(初期化)します。以下のプログラムを完成させ、*プロジェクト名*_OnAppResume関数の適切な場所に追加しましょう。

```
    // キー情報の消去
    プロジェクト名_????????(&pMe->keyPress);
    プロジェクト名_????????(&pMe->keyRelease);
    プロジェクト名_????????(&pMe->keyDown);
```

(６)キーが押された、キーが離されたという情報は、１フレームごとに消去します。定期的に消去しないと、キーの状態とフラグにズレが生じ、正しい情報が得られません。以下のプログラムを完成させ、メインループの最後に追加しましょう。

```
    プロジェクト名_????????(&pMe->keyPress);     // キープレスフラグ消去
    プロジェクト名_????????(&pMe->keyRelease);   // キーリリースフラグ消去
```

(７)キーが押されたときのイベントの処理を作成します。このイベントは、キーが押されたことを示すので、「キーが押されたフラグ」をTRUEに、「キーが押されているフラグ」もTRUEにします。
以下のプログラムを完成させ、適切な場所に追加しましょう。

```
// キープレスイベント処理
static boolean プロジェクト名_OnkeyPress(プロジェクト名* pMe, uint16 wParam, uint32 dwParam)
{
    switch(??????) {
      case AVK_UP:
        pMe->keyDown .UP     = ここは各自考えましょう;
        pMe->keyPress.UP     = ここは各自考えましょう;
        break;

      case AVK_DOWN:
        pMe->keyDown .DOWN   = ここは各自考えましょう;
        pMe->keyPress.DOWN   = ここは各自考えましょう;
        break;

      case AVK_LEFT:
        pMe->keyDown .LEFT   = ここは各自考えましょう;
        pMe->keyPress.LEFT   = ここは各自考えましょう;
        break;

      case AVK_RIGHT:
        pMe->keyDown .RIGHT  = ここは各自考えましょう;
        pMe->keyPress.RIGHT  = ここは各自考えましょう;
        break;

      case AVK_SELECT:
        pMe->keyDown .SELECT = ここは各自考えましょう;
```

```
        pMe->keyPress.SELECT = ここは各自考えましょう;
        break;

      case AVK_CLR:
        pMe->keyDown .CLR    = ここは各自考えましょう;
        pMe->keyPress.CLR    = ここは各自考えましょう;
        break;

      case AVK_0:
        pMe->keyDown .NUM0   = ここは各自考えましょう;
        pMe->keyPress.NUM0   = ここは各自考えましょう;
        break;

      case AVK_1:
        pMe->keyDown .NUM1   = ここは各自考えましょう;
        pMe->keyPress.NUM1   = ここは各自考えましょう;
        break;

      case AVK_2:
        pMe->keyDown .NUM2   = ここは各自考えましょう;
        pMe->keyPress.NUM2   = ここは各自考えましょう;
        break;

      case AVK_3:
        pMe->keyDown .NUM3   = ここは各自考えましょう;
        pMe->keyPress.NUM3   = ここは各自考えましょう;
        break;

      case AVK_4:
        pMe->keyDown .NUM4   = ここは各自考えましょう;
        pMe->keyPress.NUM4   = ここは各自考えましょう;
        break;

      case AVK_5:
        pMe->keyDown .NUM5   = ここは各自考えましょう;
        pMe->keyPress.NUM5   = ここは各自考えましょう;
        break;

      case AVK_6:
        pMe->keyDown .NUM6   = ここは各自考えましょう;
        pMe->keyPress.NUM6   = ここは各自考えましょう;
        break;

      case AVK_7:
        pMe->keyDown .NUM7   = ここは各自考えましょう;
        pMe->keyPress.NUM7   = ここは各自考えましょう;
        break;

      case AVK_8:
        pMe->keyDown .NUM8   = ここは各自考えましょう;
        pMe->keyPress.NUM8   = ここは各自考えましょう;
        break;

      case AVK_9:
        pMe->keyDown .NUM9   = ここは各自考えましょう;
        pMe->keyPress.NUM9   = ここは各自考えましょう;
        break;

      case AVK_STAR:
        pMe->keyDown .STAR   = ここは各自考えましょう;
        pMe->keyPress.STAR   = ここは各自考えましょう;
        break;

      case AVK_POUND:
        pMe->keyDown .POUND  = ここは各自考えましょう;
        pMe->keyPress.POUND  = ここは各自考えましょう;
        break;
    }

    return TRUE;
}
```

(８)(７)のプロトタイプを適切な場所に作成しましょう。

(９)キーが離されたときのイベントの処理を作成します。このイベントは、キーが離されたことを示すので、「キーが押されているフラグ」をFALSEに、「キーが離されたフラグ」をTRUEにします。(８)を参考に以下のプログラムを完成させ、適切な場所に追加しましょう。

```
// キープレスイベント処理
static boolean プロジェクト名_OnKeyRelease(プロジェクト名* pMe, uint16 wParam, uint32 dwParam)
{
    switch(??????) {


    ※case文は、問題文および(７)を参考に、各自作成してください


    }

    return TRUE;
}
```

(10)(９)のプロトタイプを適切な場所に作成しましょう。

(11)イベントハンドラでキーが押されたイベントとキーが離されたイベントが処理されるようにします。以下のプログラムを完成させ、適切な場所に追加しましょう。

```
    case EVT_???_?????:   return プロジェクト名_OnKeyPress  (pMe, wParam, dwParam);
    case EVT_???_????????: return プロジェクト名_OnKeyRelease(pMe, wParam, dwParam);
```

(12)キーの状態によって、キャラクターを移動させましょう。
・ヒント

```
    // 移動
    if(pMe->KeyDown.LEFT == TRUE)   // 左
        (左が押されている)
```

(13)キャラクターの移動範囲を画面内だけにしましょう。
・ヒント１
座標は、画像の左上を指しています。

・ヒント２
ｘ座標の最小値は、原点の0です。この値未満になるということは、原点より左側にいるということになり、画面外にいることになります。この場合は、ｘ座標を0にし、画面左に戻します。

・ヒント３
ｘ座標の最大値は、画面右端の240になりそうですが、実は違います。ｘ座標は、キャラクターの左端の座標なので、ここが240の場合は、すでに画面外に出ています。キャラクターの右端(左端 + 幅)が240を越えているかどうかを調べるか、キャラクターの左端が、240から幅を引いた値(240 - 幅)を越えているかを調べます。越えている場合は、ｘ座標を240から幅を引いた値にします。

・ヒント４
ｙ座標も同様に考えます。