ESライブラリ＆＆
ゲームプログラミング

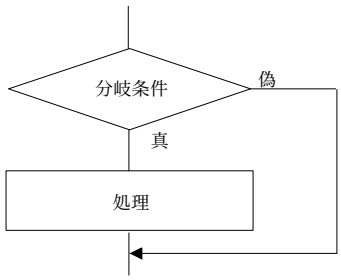
○　荷物勇者編 - 第５回　地形データで壁の描画　○

条件分岐

C／C++は、上から順番にプログラムが実行されますが、プログラムの流れを何らかの条件によって分岐させることができます。分岐は、if文やswitch文で行います。
if文は、

もし～ならば「処理」を実行し、
そうでなければ(「処理」を実行しないで)次の処理を実行する

という分岐をさせることができます。
if文の書式は以下のようになります。




```
if(条件) {
    条件を満たしたときに実行する処理
}
```

「条件」にはどのような条件でも記述できるわけではなく、基本的に２つの値を比較することしかできません。比較に使用できる演算子も決められており、以下のものしか使用できません。

==　　!=　　<　　>　　<=　　>=

「==」は等しいかどうかを調べます。「==」の左側と右側が等しいときは真(成り立つ)、等しくないときは偽(成り立たない)となります。C／C++では、「=」が代入に割り当てられているので、比較は「==」を用いるようになっています。

「!=」は「==」の逆で、「!=」の左側と右側が異なるときは真(成り立つ)、等しいときは偽(成り立たない)となります。キーボードに≠がないため、このような記述になっています。

「<」「>」は大小比較に用います。「<=」「>=」は以上と以下(キーボードに≦と≧がないためにこのように記述します)を調べる場合に用います。

課　題

地形データを作成し、それをもとに壁の描画をしましょう。

(１)地形データは文字列で定義することとします。まず、地形データを格納する文字列型の変数を宣言します。以下のプログラムをヘッダーファイル"GameMain.h"の適切な場所に追加しましょう。

```
// 地形データ
tstring    mapData;
```

C++には、文字列を扱うstring型(stringクラス)があります。tstring型は、string型をWindowsプログラミング向けにしたものです。

(２)地形データにデータを設定します。以下のプログラムをソースファイル"GameMain.cpp"の"Initialize関数"の適切な場所に追加しましょう。

```
// マップデータ初期化
MapData = TEXT( "# ##  ## #" );
```

マップデータは、'#'が壁、' '(空白)が何もないを表すものとします。

string型はその名前のとおり、文字列を扱うことを前提に作られています。string型の利点の１つに、上記のように文字列を直接代入できることが上げられます。
※宣言時、tstringをcstringにすると、TEXT()でくくらなくても文字列を代入することができます

(３)前回作成したDraw関数の「四方に壁を表示するプログラム」をコメントで囲むなどし、実行されないようにしましょう。

これから地形データにそって壁の表示を行いますが、前回作成した壁の表示プログラムがあると、今回のプログラムが正しいかどうか見分けるのが難しくなるので、実行されないようにしてください。

(４)地形データの１文字目を読み取り、座標(0.0, 0.0, 0.1)に'#'なら壁、' 'なら何も表示しないプログラムを作成します。以下のプログラムの足りない部分を補って完成させ、適切な場所に追加しましょう。

```
// 地形描画
if(mapData[0] == '#')
    SpriteBatch.Draw(*wallSpr, Vector3(※ここは各自考えてください※));
```

このプログラムによって、文字である地形データが画像として画面に表示できるようになります。

if文により条件分岐させています。mapData[0](mapDataの先頭)が'#'であれば壁スプライトを描画、そうでない場合は何もしない、という動きになります。

地形データが格納されているmapData変数は、１文字ずつ読み取ることができます。

変数名[先頭からの文字数]

とすると、特定の文字を取り出すことができます。ここで注意するのは「人間の感覚の何文字目」ではなく、「先頭を基準に何文字目なのか(先頭からの距離)」を指定しなければならないことです。
たとえば、mapDataの場合は、

先頭から０文字目(人間の感覚では１文字目)…mapData[0]　(今回の場合は'#')
先頭から１文字目(人間の感覚では２文字目)…mapData[1]　(今回の場合は' ')
先頭から２文字目(人間の感覚では３文字目)…mapData[2]　(今回の場合は'#')
先頭から３文字目(人間の感覚では４文字目)…mapData[3]　(今回の場合は'#')
⋮

となります。今回、mapDataには10文字設定したので、mapData[0]からmapData[9]まで読み取ることができます。mapData[10]は存在しないことに注意してください。

(５)(４)を参考に、mapData[1]から地形データを読み取り、座標(32.0, 0.0, 0.1)にその画像が表示されるプログラムを作成しましょう。

(６)同様に、残りmapData[2]からMmapData[9]の地形データを読み取り、座標(64.0, 0.0, 0.1)から(288.0, 0.0, 0.1)までデータにそった画像が表示されるようにしましょう。

(７)(６)までのプログラムをfor文を使った繰り返しで表示してみましょう。

地形表示プログラムは以下のようになっています。

```
if(mapData[0] == '#')       // １個目
    SpriteBatch.Draw(*wallSpr, Vector3(  0.0f, 0.0f, 0.1f));
if(mapData[1] == '#')       // ２個目
    SpriteBatch.Draw(*wallSpr, Vector3( 32.0f, 0.0f, 0.1f));
if(mapData[2] == '#')       // ３個目
    SpriteBatch.Draw(*wallSpr, Vector3( 64.0f, 0.0f, 0.1f));
                ⋮
                ⋮
if(mapData[8] == '#')       // ９個目
    SpriteBatch.Draw(*wallSpr, Vector3(256.0f, 0.0f, 0.1f));
if(mapData[9] == '#')       // 10個目
    SpriteBatch.Draw(*wallSpr, Vector3(288.0f, 0.0f, 0.1f));
```

「mapDataの[]の中」と「ｘ座標の数値」以外の部分はまったく同じです。「mapDataの[]の中」は、「１ずつ増えている」、「ｘ座標の数値」は、「32.0ずつ増えている」という法則性があります。また、

```
mapData[0] → ｘ座標 0.0 → 0 * 32.0
mapData[1] → ｘ座標32.0 → 1 * 32.0
mapData[2] → ｘ座標64.0 → 2 * 32.0
mapData[3] → ｘ座標96.0 → 3 * 32.0
```

というように、[]の中の数値とｘ座標にも関連があります。for文でまとめるのに最適なプログラムといえます。

以下のプログラムの足りない部分を補って完成させ、適切な場所に追加しましょう。

```
// 地形描画
for(int i = 0; i < 10; i++) {
    if(mapData[※ここは各自考えてください※] == '#')
        SpriteBatch.Draw(*wallSpr, Vector3(※ここは各自考えてください※, 0.0f, 0.1f));
}
```

応用問題１：地形データのサイズを増やしてみましょう。

ヒント１：mapData.size()またはmapData.length()とすると、文字数を取得することができます。ループの条件が「i < 10」の部分を書き換えると、格納した文字数分だけ繰り返しを実行することができます。

ヒント２：mapData.size()[mapData.length()も同様]は、符号なしの数値に設定されています。int型は符号ありのため、そのまま比較するとコンパイル時に警告が発せられます。
以下のように、「unsigned int型」(Windowsでは省略したUINT型でも可)にすると警告を防げます。

```
// 地形描画
for(unsigned int i = 0; i < mapData.size(); i++) {
                    ⋮
```