[バンタン体験授業(2020年03月版)]

ターン制バトルを作ってみよう

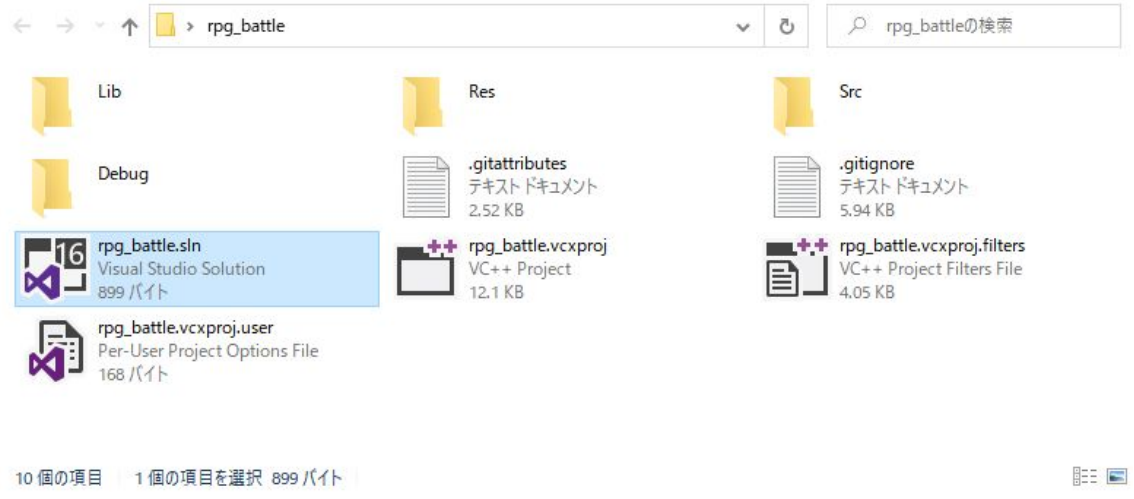
1 Visual Studioを起動する

1.1 SLNファイルを開く

このテキストでは、「C++(シー・プラス・プラス)」というプログラミング言語を使って、簡単なコンピューター・ゲームを作っていきます。また、ゲームを作るために「Visual Studio(ビジュアル・スタジオ)」というツールを使います。

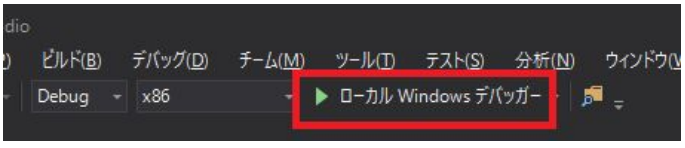
さっそくVisual Studioを起動しましょう。デスクトップの「rpg_battle(アールピージー・バトル)」というフォルダをダブルクリックして開いてください。

その中に「rpg_battle.sln(アールピージー・バトル・エス・エル・エヌ)」というファイルがあります。次はこのファイルをダブルクリックしてください。
するとVisual Studioが起動します。



1.2 プログラムを実行する

Visual Studioを起動したら、もうプログラムを実行する準備はできています。
プログラムを実行するには、上の方にある「ローカルWindowsデバッガー(ローカル・ウィンドウズ・デバッガー)」というボタンをクリックします。



しばらく待つと、プログラムが実行されてウィンドウが表示されます。
どうやら、「モンスター」が現れたようです。文章の下で「▽」が点滅していますが、これはキー操作を待っていることを示しています。

とりあえず、指示に従って何かキーボードのキーを押してみましょう。そうすると、追加のメッセージが表示されます。また「▽」が表示されるので、もう一度キーを押すとウィンドウが閉じると思います。

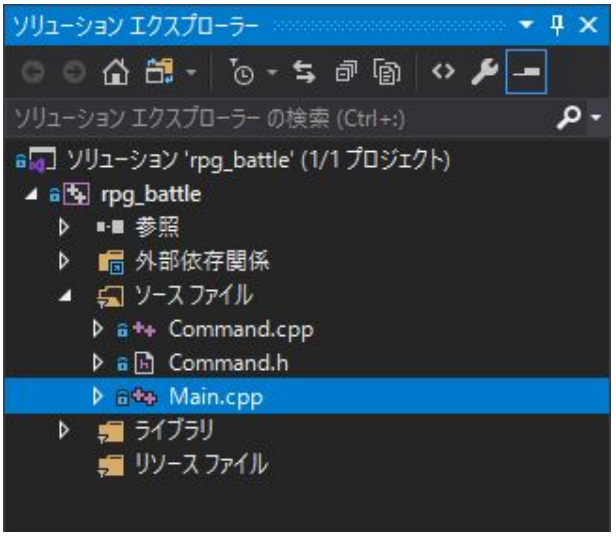
2 プログラムを見てみよう

2.1 Main.cppを開く

さきほど実行したプログラムは、どのような仕組みになっているのでしょうか。
それを調べるためには、「Main.cpp(メイン・シー・ピー・ピー)」というファイルを開かなくてはなりません。

Visual Studioの右側に「ソリューション エクスプローラー」というウィンドウがあると思います。

その中に「ソース ファイル」というフォルダアイコンがあり、さらにその下に「Main.cpp」と書かれた行があるのが分かるでしょうか。この行をダブルクリックするとMain.cppを開くことができます。



2.2 C++言語

Main.cppは「**C++(シー・プラス・プラス)**」という言語で書かれています。わたしたちが日本語を話すように、世界では国によってさまざまな言語が使われていますよね。それと同じように、コンピューターの世界にもさまざまな言語が存在するのです。
コンピューターのプログラムは「プログラミング言語」と呼ばれるもので書かれます。C++言語もプログラミング言語のひとつで、家庭用ゲームをはじめ、Windowsなど様々なソフトの開発に使わてています。

2.3 見間違えやすい文字に気をつけよう

これから、Main.cppを見ていくわけですが、その前にいくつか注意点を伝えておきます。プログラムでは**見間違えやすい文字**を使います。以下の文字は特に見間違えやすいので気をつけて読み書きしてください。

- 1(いち)、i(小文字のアイ)、I(大文字のアイ)、l(小文字のエル)、|(垂直線)
- 0(数字のゼロ)、o(小文字のオー)、O(大文字のオー)

他にも、アルファベットの大文字小文字は見間違えやすいものです。注意して読むようにしてください。なお、このテキストで使われているのはほとんどが小文字です。

2.4 プログラムを書くときは「半角モード」を使おう

パソコンで日本語を入力するには、キーボード左上にある「半角/全角(はんかく/ぜんかく)」キーを押して「**全角モード**」にします。日本語の入力を漢字に変換するには、キーボード中央下にある細長い「**スペースバー**」を押します。押すたびに変換候補が切り替わります。キーボード右側の「**Enter**(エンター)」キーを押すと、表示された漢字が決定されます。もう一度「半角/全角キー」を押すと「**半角モード**」に戻ります。

C++のプログラムを書くときは、常に**半角モード**を使います。日本語を使えるのは、基本的にはコメントと文章の部分だけです。うっかり全角モードのままプログラムを書いてしまったときは、落ち着いてEnterキーの上の「**backspace**(バックスペース)」キーを押して全角モードの文字を消し、半角モードで書きなおしましょう。

2.5 インクルード指令

```
1 #include "Command.h"
```

プログラムの先頭にある「**#include**(シャープ・インクルード)」は、**インクルード指令**といいます。インクルード指令を使うと、プログラムで使う部品を追加することができます。

実は、現在のC++言語には、画像や音声を扱う機能がありません。そこで、このテキストのために、それらを扱う部品を追加しています。「Command.h」(コマンド・ドット・エイチ)というのが、その部品の書かれたファイルの名前です。

Command.hには、この「ターン制バトルゲーム」で使うための、さまざまな部品が含まれています。

2.6 イント・メイン

```
3 // ここからプログラムの実行が開始される
4 int main()
5 {
```

C++プログラムは「**int main**(イント・メイン)」から始まります。mainのすぐ後ろには「(」(丸かっこ)と「)」(閉じ丸かっこ)があります。プログラムに受け渡すデータ

がある場合、それをこの丸かっこの内側に書きます。今回は渡すデータがないので何も書いていません。

その次の行の「{」(波かっこ)から、ファイルの一番下の「}」(閉じ波かっこ)までが実行されるプログラムの内容になっています。つまり、この波かっこの内側に書いてあるものが「ターン制バトルゲームのプログラム」ということです。**プログラムは上の行から順番に実行されます。**

3行目の緑色の部分は「**コメント**」といいます。コメントは人間がプログラムを読みやすくするためのものです。コンピューターはコメントを無視します。コメントは「**//**(ダブル・スラッシュ)」ではじまり、改行で終わります。

2.7 プログラムの初期化

```
 6|   // プログラムの初期化処理
 7|   initialize("モンスター・バトル");
```

「**initialize**(イニシャライズ)」は、Command.hに含まれている部品をセットアップして、プログラムで使える状態にします(C++標準の機能ではありません)。丸かっこと閉じ丸かっこの内側に書かれているのは「ウィンドウ・タイトル」です。これは、プログラムを実行したとき、ウィンドウの上枠に表示される文章です。

ところで、この文章は「**"**(ダブル・クォーテーション)」という記号でかこんでいます。C++では、文章を書きたいときは、ダブル・クォーテーションで囲むことになっているからです。

行の最後には「**;**(セミコロン)」という記号があります。C++言語では、行の終わりにセミコロンを付ける決まりになっています。セミコロンはとても重要です。行の終わりには忘れずに付けるようにしましょう。

2.8 画像を表示する

```
 9|   // 背景を表示
10|   image background;
11|   background.set(400, 300, "bg_paper.jpg");
```

10～11行目は背景画像を表示しています。10行目の先頭には「**image**(イメージ)」という単語があります。これは画像を取り扱う「**クラス**」の名前です。

クラスというのは、**特定の処理をひとまとめにして、そのかたまりに分かりやすい名前を付けたもの**、と考えてください。imageクラスを使うと、画面に画像を表示することができます。

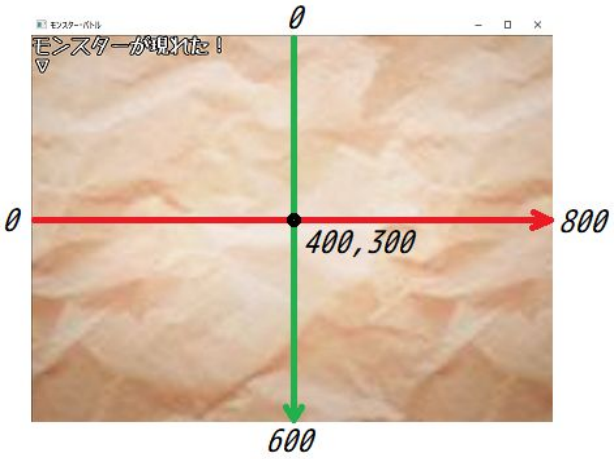
先に説明したように、C++には画像を扱う機能はありません。しかし、こうして独自のクラスを作ることで、**機能を拡張することができる**のです。

空白をはさんで、「background(バックグラウンド)」という単語があります。これは、画像を操作するときの名前になります。たくさんの画像を表示すると、クラス名だけでは操作したい画像を見分けられなくなってしまいます。こうして、**画像ごとに異なる名前を付けることで、見分けがつくようにしている**のです。

この別名は、さっそく11行目で登場します。「.(ドット)」記号の次にある「set(セット)」というのが、操作の内容を表しています。setは「画像をセットする」という操作を表します。

その後ろに丸括弧が続きます。丸括弧の内側には「,(カンマ)」で区切られた3つのパラメーターがあります。

最初の2つのパラメーターは「400」と「300」で、これは画像を表示する位置を表します。ひとつめがウィンドウの左端からのピクセル数、ふたつめが上端からのピクセル数です。



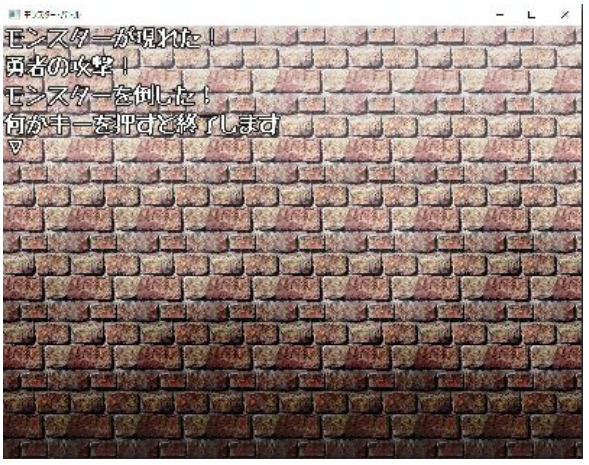
そして、ウィンドウの大きさは横が800、縦が600で、原点は左上にあります。つまり、座標(400,300)というのは、**ウィンドウの中心**にあたるわけです。

最後の「"bg_paper.jpg"」は、表示する画像ファイルの名前です。**C++では、ファイル名は文章として扱われます。**だから「"」で囲まなくてはなりません。

ここで少し、プログラムをいじってみましょう。ゲームの雰囲気を変えるために、背景を変えてみます。11行目のファイル名を、次のように書き換えてください。

```
10|  image background;
11|  background.set(400, 300, "bg_brick.jpg");
```

文章を書き換えたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。レンガの背景が表示されていれば成功です。キーを押してプログラムを終了させてください。



2.9 文章を表示する

```
13|   // 文章を表示
14|   printf("モンスターが現れた！¥n");
```

Main.cppに戻ります。14行目では画面に文章を表示しています。「**printf**(プリント・エフ)」を使うと、丸括弧の内側に書かれた文章が画面に表示されます。

文章の最後にある「**¥n**(エン・エヌ)」は、改行を表します。

2.10 キーが押されるのを待つ

```
16|   // 何かキーが押されるのを待つ
17|   wait_any_key();
```

「**wait_any_key**(ウェイト・エニー・キー)」を使うと、画面に「▽」を点滅させて入力を待つことができます(これはCommand.hで追加した機能です)。

2.11 文章を何行も表示する

```
19|   // 勇者のターン
20|   printf("勇者の攻撃！¥n");
21|   printf("モンスターを倒した！¥n");
22|   printf("[何かキーを押すと終了します]¥n");
23|   wait_any_key();
```

printfを続けて書くと、書いたぶんだけ文章が表示されます。改行したいときは「¥n」を書きます。

2.12 プログラムの終了

```
25|   // プログラムの終了処理
26|   finalize();
```

プログラムの最後にある「**finalize**(ファイナライズ)」は、initializeでセットアップした部品を解体し、プログラムを安全に終了できるようにします(これはCommand.hで追加した機能です)。

3 モンスターを表示しよう

3.1 モンスターの画像を表示する

文章では「モンスターが現れた！」と言っているのに、画面には何も表示されていません。これでは何が起こったのか全くわかりませんよね。そこで、画面にモンスターの画像を表示しましょう。

15～17行目に、「imageクラスを使ってスライムの画像を表示する」プログラムを書き加えてください。

```
14  printf("モンスターが現れた！¥n");
15
16  image monster;
17  monster.set(400, 400, "slime.png");
18
19  // 何かキーが押されるのを待つ
20  wait_any_key();
```

画像の表示にはimageクラスを使うのでした。モンスター用の画像ということで、「monster(モンスター)」という名前を付けました。

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。スライムの画像が表示されていたら成功です。

3.2 モンスターの画像を消す

今度は「モンスターを倒した！」と表示するタイミングで画像を消しましょう。画像を消すには「**clear**(クリア)」という操作をします。24行目に次のプログラムを書き加えてください。

```
23  printf("勇者の攻撃！¥n");
24  printf("モンスターを倒した！¥n");
25  monster.clear();
26  printf("[何かキーを押すと終了します]¥n");
27  wait_any_key();
```

「clear」はsetと同じく「imageクラス」の操作の一つです。

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。「モンスターを倒した！」と表示されたとき、スライムの画像が消えたら成功です。

【課題１】
モンスターの画像を「goblin.png」に変更しなさい。「Res/画像」フォルダにある画像から、好きなものを使ってもいいでしょう。

4 モンスターのHP(ヒットポイント)を作ろう

4.1 数値に名前をつける

今のところ、モンスターは一撃でやられてしまいます。敵として不甲斐ないので、体力を表すHPを設定して、簡単にはやられないようにしましょう。

HPのような整数を扱うには「**int**(イント)」というクラスを使います。「int」は「integer(インテジャー、「整数」という意味)」という単語を省略したものです。intは次のように書きます。

```
int 名前 = 数値;
```

imageクラスのときと同じく、名前を付けることで、それが何のための数値なのかが分かるようにします。名前の後ろに「=(イコール)」が続き、その次に名前が表す数値を書きます。このように数値が代入された名前のことを**変数**と呼びます。

それではHPを作りましょう。モンスターのHPなので、名前は「monster_hp(モンスター・エイチピー)」とします。数値はとりあえず7にしましょう。それでは、15～16行目に次のプログラムを書き加えてください。

```
14    printf("モンスターが現れた！¥n");
15
16    int monster_hp = 7;
17
18    image monster;
19    monster.set(400, 400, "slime.png");
```

《コラム》変数が使える範囲
変数は、その名前をつけた行より下でしか使えません。
人間と同じように、コンピューターも、まだ読んでいない部分に書かれていることは分からないのです。

4.2 数値を減らす

次は、HPを減らす処理を追加します。C++では以下の式で数値の計算ができます。

```
名前 = 式;
```

「式」の部分は算数や数学の式と同じように書きます。例えば「100+100」や「5-(10+1)」などです。また、数値のかわりに名前を使うこともできます。

C++言語の「=」には、数学の「=」にある「等しい」という意味が**ない**ことに気をつけてください。数学の「=」には「等しい」と「代入」という２つの意味があり、書いてある場所や前後の文章からどちらなのか判断します。しかし、**C++言語の「=」には「代入」という意味しかありません。**だから、C++言語では、以下のように書くことが可能です。

```
monster_hp = monster_hp - 3;
```

算数や数学ではおかしな式ですが、C++では「monster_hpから3を引いた値をmonster_hpに代入する」という意味になるので問題ありません。それでは、25行目に次のプログラムを書き加えてください。

```
24|   // 勇者のターン
25|   printf("勇者の攻撃！¥n");
26|   monster_hp = monster_hp - 3;
27|   printf("モンスターを倒した！¥n");
```

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。しかし、やっぱりモンスターは一撃でやられてしまいます。

【課題２】

モンスターのHPを減らすプログラムのすぐ下に、printfで「モンスターに3のダメージ！」という文章を表示するプログラムを書きなさい(ダブル・クォーテーションは「Shiftを押しながら2キーを押す」と出ます)。
「ローカルウィンドウズデバッガー」をクリックして、実際に表示されるのを確認すること。

《コラム》計算に使う記号

C++言語の足し算と引き算は、算数や数学と同じく「+」と「-」を使います。しかし、掛け算は「×」のかわりに「＊」(アスタリスク、星印)、割り算は「÷」のかわりに「/」(スラッシュ、斜線)を使います。これは、古いコンピューターは性能が低くて、掛け算や割り算の記号が使えなかったことが原因です。
そのため、当時のプログラミング言語の開発者たちは、掛け算には、数学の乗算記号である「・」(ドット、中点)に見えなくもない「＊」を、割り算には、分数に見える「/」を使うことに決めました。
それがC++言語にも受け継がれているのです。

4.3 残りHPを調べる

HPが残っているのにやられてしまうのは、どれだけHPが残っているかを調べていないからです。残りHPを調べるには「**if**(イフ、「もしも～ならば」という意味)」を使います。ifは次のように書きます。

```
if (条件) {
  条件が満たされたときだけ実行するプログラム
}
```

ifに続く「()」の内側に条件を書きます。**条件が成立したときだけ、そのあとの「{}」の内側にあるプログラムが実行されます。**不成立の場合は「{}」の内側は全く実行されません。

これを踏まえて、モンスターを即死させないためには、まず「HPが0以下」かどうかを条件とします。そして、「モンスターを倒した！」という文章の表示と、モンスターの画像を消す処理を、「条件が成立したときだけ実行される」ようにしていきます。では、モンスターのHPを減らすプログラムの下に、以下のプログラムを書き加えてください。

```
25|   printf("勇者の攻撃！¥n");
26|   monster_hp = monster_hp - 3;
27|   printf("モンスターに３のダメージ！¥n");
28|   if (monster_hp <= 0) {
29|     printf("モンスターを倒した！¥n");
30|     monster.clear();
31|   }
32|   printf("[何かキーを押すと終了します]¥n");
33|   wait_any_key();
```

「ある数値以下」という条件は、数学では「≦」と書けます。しかし、C++にはこの記号がないので、代わりに「<」と「=」を横につなげて「<=」のように書きます。このとき、**2つの記号の間に空白を入れない**ように注意してください。

条件として使える記号を以下に示します。

意味	C++での書き方
小なりイコール(≦)	<=
大なりイコール(≧)	>=
小なり(<)	<
大なり(<)	>
等しい(=)	==
等しくない(≠)	!=

なお、**C++で「等しい」を表すときは、イコールを2つ書かなくてはなりません。**なぜなら、イコール１つは「代入」を表すからです。これは忘れやすいので、しっかり覚えておきましょう。

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。モンスターが死ななければ成功です。

5 勇者のHPを作ろう

5.1 攻撃に耐えるために

モンスターは勇者の攻撃に耐えて生き残りました。今度はモンスターのほうが勇者を攻撃してくるでしょう。しかし、HPがなければ、その攻撃に耐えることができません。

そこで、勇者にもHPを作りましょう。次のように、モンスターのHPの上に、「hero_hp(ヒーロー・エイチピー)」という名前で、勇者のHPを表す変数を作成してください。

```
14  printf("モンスターが現れた！¥n");
15
16  int hero_hp = 20;
17  int monster_hp = 7;
18
19  image monster;
20  monster.set(400, 400, "slime.png");
```

これで、モンスターの攻撃に耐えることができるでしょう。

5.2 モンスターに攻撃させる

それでは、モンスターに攻撃してもらいましょう。モンスターの死亡判定の下に、次のプログラムを書き加えてください。

```
29  if (monster_hp <= 0) {
30    printf("モンスターを倒した！¥n");
31    monster.clear();
32  }
33
34  // モンスターのターン
35  printf("モンスターの攻撃！¥n");
36  hero_hp = hero_hp - 2;
35  printf("勇者は２のダメージ！¥n");
36  printf("[何かキーを押すと終了します]¥n");
37  wait_any_key();
```

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。勇者の攻撃のあと、モンスターが攻撃してきたら成功です。

6 モンスターを倒そう

6.1 プログラムを繰り返し実行する

モンスターは一撃で死ななくなりました。しかし、一度しか攻撃していないので、モンスターを倒すことができません。そこで、モンスターを倒すまで繰り返し攻撃するようにしましょう。

プログラムを繰り返し実行するには「**for**(フォー)」を使います。forは次のように書きます。

```
for (;;) {
  繰り返し実行するプログラム
}
```

繰り返したい範囲の手前に「for (;;) {」と書き、最後に「}」を書くと、その範囲にあるプログラムは、**無限に繰り返し実行**されるようになります。

このforを使い、勇者の攻撃と、モンスターのHP判定を、繰り返し実行するようにしましょう。それでは、勇者とモンスターのターンを囲うように、forを書き加えてください。

```
22|   // 何かキーが押されるのを待つ
23|   wait_any_key();
24|
25|   for (;;) {
26|     // 勇者のターン
27|     printf("勇者の攻撃！¥n");
28|     monster_hp = monster_hp - 3;
29|     printf("モンスターに３のダメージ！¥n");
30|     if (monster_hp <= 0) {
31|       printf("モンスターを倒した！¥n");
32|       monster.clear();
33|     }
34|
35|     // モンスターのターン
36|     printf("モンスターの攻撃！¥n");
37|     hero_hp = hero_hp - 2;
38|     printf("勇者に２のダメージ！¥n");
39|   }
40|   printf("[何かキーを押すと終了します]¥n");
41|   wait_any_key();
```

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。何度か攻撃が行われ、モンスターを倒すことができたら成功です。**プログラムを終わらせるには、ウィンドウ右上の「×」ボタンをクリック**してください。

6.2 繰り返しを終了する

繰り返し攻撃することでモンスターを倒せるようにはなりました。しかし、今度はいつまでも倒し続けるようになってしまいました。こうなってしまうのは、繰り返しを終わらせるプログラムを書いていないからです。こうした終わりのない繰り返しのことを「**無限ループ**」といいます。

無限ループを終わらせるには「**break**(ブレイク)」を使います。breakが実行されると、**forの最後の「}」までのプログラムは飛ばされ、繰り返しが終了**します。今回は、敵を倒したときに終わらせるようにします。次のように、敵を倒した直後にbreakを加えてください。

```
30      if (monster_hp <= 0) {
31        printf("モンスターを倒した！¥n");
32        monster.clear();
33        break;
34      }
35
36      // モンスターのターン
37      printf("モンスターの攻撃！¥n");
```

これで、無限ループではなくなります。プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。モンスターを倒したあと、ゲームを終了できたら成功です。

6.3 勇者がやられることもある

今の勇者は不死身です。というのも、モンスターのように残りHPを調べていないからです。自らの死に気づかず戦うことを止めない狂戦士…などというとかっこいいですが、ゲームとしてはちょっとまずいですね。

まず、実際にHPが減っていることが分かるように、HPを表示しましょう。整数の値を表示するには「**%d**(パーセント・ディ)」を使います。**printfの文章中に%dを書き、追加のパラメーターとして、表示したい変数を加えます。**

勇者のターンの先頭に、次のプログラムを書き加えてください。

```
25    for (;;) {
26      // 勇者のターン
27      printf("勇者のHP=%d¥n", hero_hp);
28      printf("勇者の攻撃！¥n");
29      monster_hp = monster_hp - 3;
```

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。勇者のHPが表示されていたら成功です。

HPの様子が分かるようになったら、今度は死亡判定を追加します。モンスターの攻撃のあとに、勇者のHPを調べる処理を追加してください。

```
38|     printf("モンスターの攻撃！¥n");
39|     hero_hp = hero_hp - 2;
40|     printf("勇者に２のダメージ！¥n");
41|     if (hero_hp <= 0) {
42|       printf("勇者は死んでしまった！¥n");
43|       break;
44|     }
45|   }
46|   printf("[何かキーを押すと終了します]¥n");
```

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。見た目には違いはないようですが、まあ、勇者は簡単にやられたりはしないものです。

しかし現実は厳しいのです。次の課題を終了させて、勇者に現実を教えてあげてください。

【課題３】
勇者が一度に受けるダメージを10にして、勇者が不死身でなくなったことを確認しなさい。勇者に現実を思い知らせたら、ダメージを2に戻しなさい。

6.4 一定時間待つ

とりあえず戦いらしくなってきましたが、文章が一気に表示されるため「気づいたときには戦いが終わっている」という状態です。臨場感に欠けるので、文章を読む時間を作ろうと思います。

「**usleep**(ユー・スリープ)」を使うと、**指定したマイクロ秒だけプログラムの実行を停止**できます(1秒=1,000,000マイクロ秒)。巨大な数値は桁数が分かりにくいので、C++では任意の位置に「'(シングル・クォーテーション、shiftを押しながら7)」を入れられます。では、勇者のターンに、次のプログラムを書き加えてください。

```
27|     printf("勇者のHP=%d¥n", hero_hp);
28|     printf("勇者の攻撃！¥n");
29|     usleep(500'000); // 0.5秒待つ
30|     monster_hp = monster_hp - 3;
31|     printf("モンスターに３のダメージ！¥n");
32|     usleep(1'000'000); // 1秒待つ
33|     if (monster_hp <= 0) {
34|       printf("モンスターを倒した！¥n");
```

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。文章が少しずつ表示されたら成功です。待ち時間が短い、あるいは長いと感じたら、長さを調整するとよいでしょう。

【課題４】
usleepを使って、モンスターのターンでも、文章が少しずつ表示されるようにしなさい。

6.5 文章を消去する

戦闘が長引くと、文章が画面に収まらなくなってきます。あまり見栄えがよいとは言えませんね。そこで、ターンが始まるタイミングで、以前の文章を消去することにします。

ただ、C++言語には標準的な画面消去の仕組みがありません。そこで、このテキストのために「**clear_text_all**(クリア・テキスト・オール)」という機能を用意しました。

勇者のターンが始まる前に、次のプログラムを書き加えてください。

```
25   for (;;) {
26     clear_text_all(); // 文章を消す
27     // 勇者のターン
28     printf("勇者のHP=%d¥n", hero_hp);
```

プログラムが書けたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。ターンが始まるたびに、上から文章が書き直されていたら成功です。

7 戦うか逃げるか、それが問題だ

7.1 選択肢を表示する

現在のプログラムは、自動的に攻撃をしています。敵が勇者より弱いならそれで構いわないでしょう。しかし、今の勇者には手に負えない、強い敵に出会ってしまったとしたら…?

そんなときのために、戦うか、逃げるかを選択できるようにしましょう。このテキストのために、選択肢を表示する「**select**(セレクト)」という機能を用意しましたので、これを使っていきます。

selectを使うと、選択肢を表示して勇者に選ばせることができます。selectは以下のように書きます。

```
int 出力先の変数 = select(選択肢の数, 選択肢0の文章, 選択肢1の文章, ...);
```

selectは数学の関数のように、入力パラメーターに対応する数値を出力します。丸括弧の内側が入力パラメーターです。最初のパラメーターは**選択肢の数**で、残りのパラメーターは**選択肢として表示される文章**です。出力は**選ばれた選択肢の番号**です。

勇者のHPを表示するprintfの下に、次のプログラムを書き加えてください。

```
28      printf("勇者のHP=%d¥n", hero_hp);
29      int hero_action = select(2, "戦う", "逃げる");
                                  1は「逃げる」選択肢の番号だよ
30      if (hero_action == 1) {
31        printf("勇者は逃げ出した！¥n");
32        break;
33      }
34      printf("勇者の攻撃！¥n");
35      usleep(500'000); // 0.5秒待つ
```

追加したプログラムは、

> selectで選択肢1、つまり「**逃げる**」が選ばれた場合に戦闘から逃げ出す

というものです。

書き加えたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。選択肢が表示されるので、矢印キーの上下で「逃げる」を選び、Enterキーで決定してください。無事に戦闘から逃げられたら成功です。

7.2 モンスターに行動を選ばせる

モンスターのほうも、いろいろな行動をするでしょう。とりあえず、気まぐれで様子を見る行動をしてもらうことにしましょう。

このようなときは、正の整数を無作為に出力する**rand**を使います。モンスターのターンに次のプログラムを書き加えてください。

```
45      // モンスターのターン
46      int monster_action = rand() % 3;
47      // 余りが1以下なら攻撃、2なら様子を見る
48      if (monster_action <= 1) {
49        printf("モンスターの攻撃！¥n");
50        usleep(500'000);
51        hero_hp = hero_hp - 2;
52        printf("勇者に２のダメージ！¥n");
53        usleep(1'000'000);
54      }
55      if (hero_hp <= 0) {
56        printf("勇者は死んでしまった！¥n");
```

randが出力する数値の範囲はとても広いので、そのままでは使いにくいです。そこで、「余り」を求めることで思い通りの範囲の数値を求めます。「余り」を求めるには「**%**(パーセント)」記号を使います。

今回は、1/3の確率で様子を見るようにしたいです。逆に言うと2/3の確率で攻撃するわけです。そこで、3で割った余りを計算し「**余りが1以下のときは攻撃する、2のときは様子を見る**」としました。

書き加えたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。何度かプログラムを実行して、たまにモンスターが何もしてこなければ成功です。

【課題５】
monster_actionが2のとき、「モンスターは様子を見ている」という文章を表示しなさい。

8 戦いの音を鳴らそう

8.1 効果音を鳴らそう

臨場感を出すために、音声を鳴らしましょう。しかし、C++言語に音声を鳴らす機能はありません。そこで、このテキストのために音声再生機能を用意しておきました。

効果音を鳴らすには「play_sound(プレイ・サウンド)」を使います。play_soundは、指定された音声を1回だけ再生します。また、**play_soundを連続して書くと、全て同時に再生**されます。

それでは、勇者の攻撃の文章を表示するプログラムの下に、次のプログラムを書き加えてください。

```
31        printf("勇者は逃げ出した！¥n");
32        break;
33      }
34      printf("勇者の攻撃！¥n");
35      play_sound("attack-sword.mp3");
36      usleep(500'000); // 0.5秒待つ
37      monster_hp = monster_hp - 3;
```

書き加えたら「ローカルWindowsデバッガー」ボタンをクリックして実行…の前に、音量をチェックしましょう。音量は画面右下のスピーカーアイコンをクリックすると変更できます。

音量を調整して、大丈夫そうならプログラムを実行してください。攻撃したとき、剣を振る音が再生されたら成功です。

8.2 BGMを鳴らそう

次はBGMを鳴らしてみましょう。BGMを鳴らすには「play_bgm(プレイ・ビージーエム)」を使います。

play_bgmは**指定された音声を無限に繰り返し再生**します。もう一度play_bgmを実行すると、**再生されていたBGMは停止して、新しいBGMが再生されます。**

初期化を行うプログラムの下に、次のプログラムを書き加えてください。

```
 6|   // プログラムの初期化処理
 7|   initialize("モンスター・バトル");
 8|
 9|   play_bgm("bgm-the-whip.mp3");
10|
11|   // 背景を表示
12|   image background;
```

書き加えたら「ローカルWindowsデバッガー」ボタンをクリックして実行してください。BGMが再生されたら成功です。

【課題６】
play_soundを使って、モンスターが攻撃するときに効果音を鳴らしなさい。
効果音は「Res/音声」の中にあります。

【課題７】
play_bgmを使って、モンスターを倒したときにBGMを切り替えなさい。
BGMは「Res/音声」の中にあります。

9 祝🎉　ゲーム完成！

おめでとうございます！
これでターン制バトルゲームのテキストは終了です。お疲れ様でした。

あとは、**あなたの好きなようにゲームを改造してください。**
ダメージをランダムにしたい？　randと%dを使ってください。
クリティカルヒットを出したい？　randとifを使ってください。
魔法をや道具を使いたい？　selectに選択肢を追加して、ifで分岐してください。

他にもやりたいことがあれば、**いつでも質問してください。**

https://github.com/tn-mai/rpg_battle