[バンタン体験授業(2024年09月版)]

じゃんけんゲームを作ってみよう

1 Visual Studioを起動しよう

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。

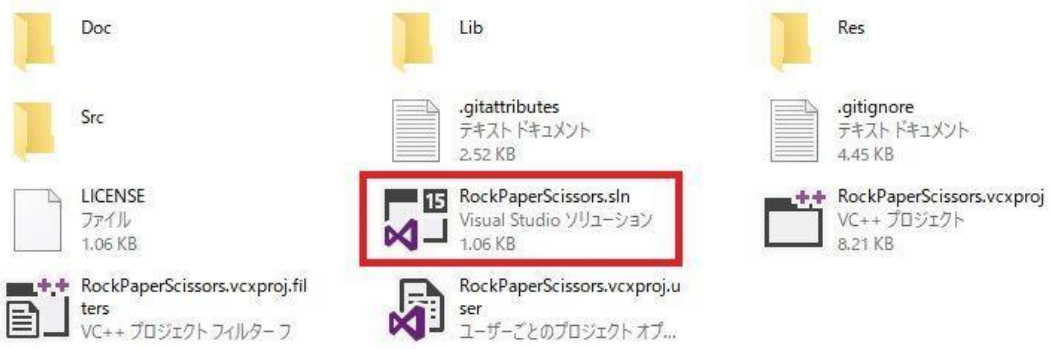
さっそくVisual Studioを起動しましょう。デスクトップの

RockPaperScissors(ロック・ペーパー・シザーズ)

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、

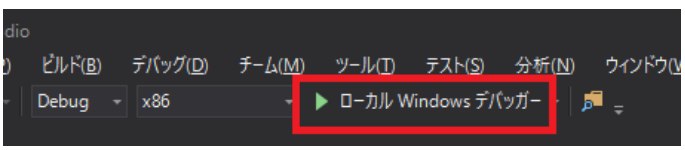
RockPaperScissors.sln(ロック・ペーパー・シザーズ・エス・エル・エヌ)

というファイルがありますので、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。



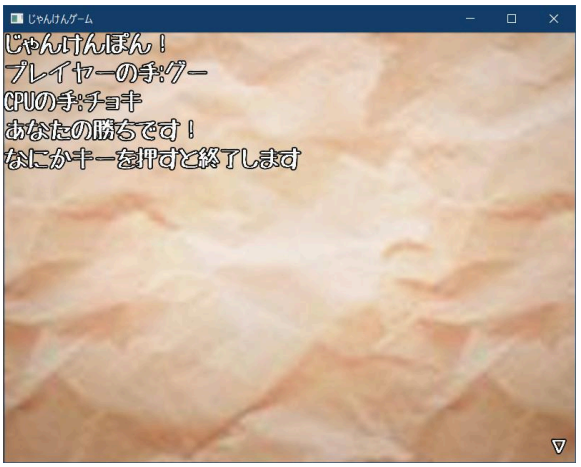
1.2 プログラムを実行する

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、じゃんけんが行われて勝利したようですね。

よく分かりませんが、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、ウィンドウが閉じると思います。



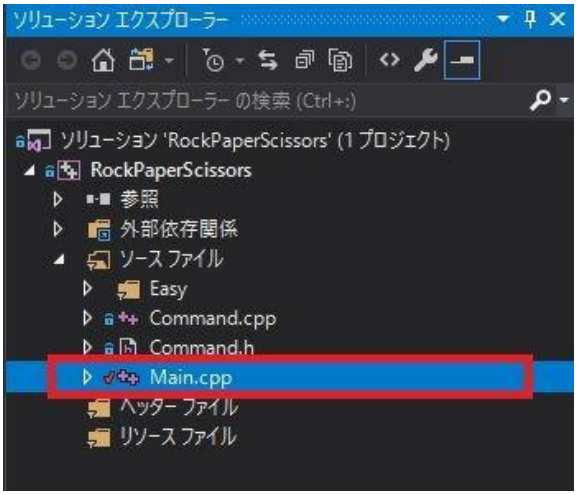
2 プログラムを見てみよう

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ます。

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。(**見つからないときは、項目の左側にある小さな「▷」アイコンをクリックしてみてください**)。



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。
「プログラミング言語」はコンピューターのプログラムを書くための言語です。
C++言語は、以下の特徴を備えています。

- コンピューターの性能を効率的に利用できる。
- 大規模なソフトウェアを効率的に開発することに適している。
- 前身となるC言語の資産を簡単に利用できる。

これらの特徴から、家庭用ゲームやWindowsなど様々なソフトウェアの開発に使われています。

Main.cppを見ていく前に、ひとつ注意点をお伝えします。
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んでください。

- 1(いち)、i(小文字アイ)、I(大文字アイ)、l(小文字エル)、|(垂直線)
- 0(数字のゼロ)、o(小文字のオー)、O(大文字のオー)
- .(ドット)と,(カンマ)
- :(コロン)と;(セミコロン)

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。

2.3 プログラムを書くときは「半角モード」を使おう

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++のプログラムで日本語を使えるのは、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て、半角モードで書きなおしましょう。

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。

2.4 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。

「**Command.h**(コマンド・ドット・エイチ)」という部分が、部品の書かれたファイルの名前です。Command.hには、この「じゃんけんゲーム」で使うことができる、さまざまな部品が含まれています。

実は、**C++**言語には画像や音声を扱う機能がありません。画像を表示したり音声を再生するには、C++言語からOS(Windowsなど)の機能を呼び出す必要があります。

しかし、体験授業の時間では、OSの機能まで呼び出している時間はありません。
そこで、Command.hの中にOSの機能を簡単に呼び出すための部品を用意しています。

それから、ファイル名を「"(ダブル・クォーテーション)」という記号でかこんでいます。これは、以下のC++のルールがあるからです。

ファイル名や文章を書くときは、"(ダブル・クォーテーション)で囲みます。

2.5 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「じゃんけん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。コメントは「**//**(スラッシュ・スラッシュ)」ではじまり、改行で終わります。

コンピューターはコメントを無視します。

2.6 プログラムの初期化と終了

```
6|   // プログラムの初期化処理
7|   initialize("じゃんけんゲーム");
```

「**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。かっこの内側に書かれているのは、プログラムを実行して表示されたウィンドウの上枠に表示された文章です。

コンピューターは、CPPファイルに書いてあるものは全てプログラムだと考えます。そのため、ただ「じゃんけんゲーム」と書くだけだと、「じゃんけんゲームという名前の部品」を探し始めます。

当然ですが、そんな部品が見つかるはずはありません。そのため、エラーになってしまいます。部品ではなく文章として扱ってほしい場合は、「"じゃんけんゲーム"」のように「"(ダブル・クォーテーション)」で囲まなくてはなりません。

最後の「**;**(セミコロン)」は行の終わりを示します。

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。

```
37  // プログラムの終了処理
38  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。

2.7 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(クラス名)です。

「**クラス**」というのは、**いくつかの処理をひとまとめにして、分かりやすい名前を付けたもの**です。**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。

すでに説明したように、C++には画像を扱う機能がありません。しかし、このように独自のクラスを作ることで、機能を拡張できるのです。

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。この部分は「**変数**」といって、画像を操作するときの名前(変数名)になります。

つまり、「**image**クラスの機能を持つ**background**(という名前の)変数」を作成しているわけです。

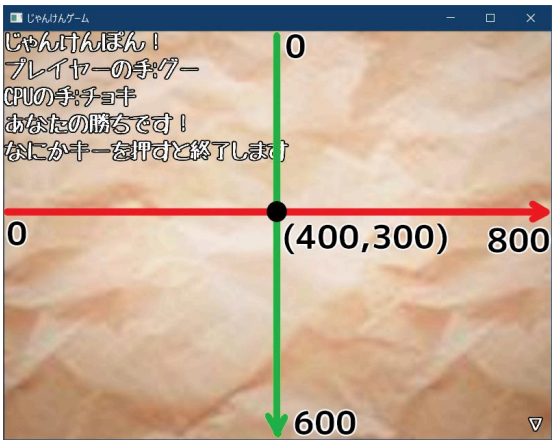
変数を作る理由は、複数の画像を扱えるようにするためです。画像ごとに異なる変数を作ることで、それぞれの画像を操作できるのです。

11行目では、この**background**変数を使ってクラスの機能を実行しています。「**.**(ドット)」記号の次の「**set**(セット)」というのが機能の名前(機能名)です。**set**の機能は「画面に画像をセットする」というものです。

機能名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**」です。

これは、**画像を表示する位置**を表します。１つめがウィンドウの左端からのピクセル数、２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。

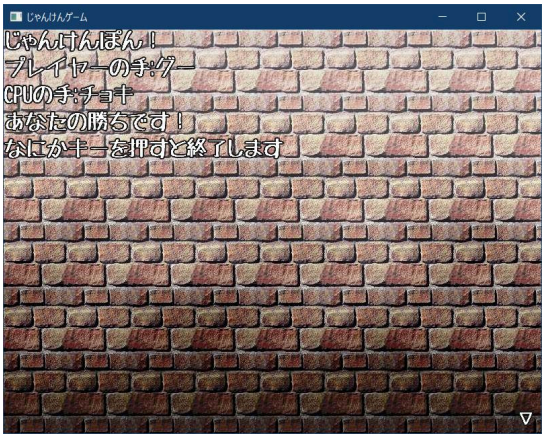


最後の「**"bg_paper.jpg"**」は、**表示する画像ファイルの名前**です。C++のファイル名は文章と同じ扱いなので、「"」で囲みます。

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。10行目のファイル名を、次のように書き換えてください。(書き換えたり追加する行は、左端の行番号が赤色になっています)。

```
10   // 背景を表示
11   set_image(No_0, 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。



2.8 文章を表示する

```
13   // 文章を表示
14   printf("じゃんけんぽん！");
```

「**printf**(プリント・エフ)」を使うと、画面に文章を表示できます。丸括弧の内側に書かれた文章が画面に表示されます。

2.9 数値に名前を付ける

```
16   // じゃんけんの手を定義する
17   int gu = 0;
18   int choki = 1;
19   int pa = 2;
```

16～18行目は、じゃんけんの手に数値を割り当てています。

じゃんけんでは「グー」、「チョキ」、「パー」の3種類の手で勝敗を決めます。ところが、コンピューターというのは数字の扱いは得意なのに、文字の並びを扱うのは不得意です。そこで、それぞれの手に番号を割り当ててあげます。

番号を使うとコンピューターは助かりますが、今度は人間のほうが理解しにくくなってしまいます。人間もコンピューターも理解しやすいプログラムが書けるように、C++言語には「**int**(イント)」という整数を表すクラスが用意されています。

intクラスを使って「整数に名前をつける」には、次のように書きます。

```
int 名前 = 数値;
```

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も**変数**といいます。

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なります。

名前を付けたことで、コンピューターはプログラムに**gu**と書かれていたら、それを数値の**0**として認識するようになります。同様に**choki**と書けば**1**、**pa**と書けば**2**と認識します。

変数が使える範囲

名前は、名前をつけた行より下側でしか使えません。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。

2.10 数値に名前をつける(その2)

```
21  // プレイヤーの手を決める
22  int player_hand = gu;
23
24  // コンピューターの手を決める
25  int cpu_hand = choki;
```

21行目と24行目は、それぞれプレイヤーの手とコンピューターの手に名前を付けています。「数値」の部分には、16～18行目で付けた名前を使っています。

2.11 条件によって違うことをする

```
27  // プレイヤーの手を表示する
28  if (player_hand == gu) {
29    printf("プレイヤーの手:グー");
30  }
31
32  // コンピューターの手を表示する
33  if (cpu_hand == choki) {
34    printf("CPUの手:チョキ");
35  }
```

28～30行目では、プレイヤーの手を表示しています。33～35行目では、コンピューターの手を表示しています。手は3種類あるので、選んだ手だけを表示しなくてはなりません。このような、「ある条件のときだけ何かをしたい」場合は、

if(イフ、「もしも～ならば」という意味)

を使います。**if**に続く「**()**」の内側に条件を書きます。条件が成立したときに限り、そのあとに続く「**{}**」の内側にあるプログラムが実行されます。

数値が同じかどうかを調べるには「**==**(イコール・イコール)」記号を使います。

「player_hand == gu」という部分が「条件」です。この条件の意味は、「プレイヤーの手がグーと等しい」です。まとめると、28～30行目の内容は、

プレイヤーの手がグーのときだけ、画面に「グー」と表示する。

という意味になります。

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(≠)	!=

ということは、プレイヤーの手をチョキやパーに変えたら、「グー」は表示されなくなるはずです。試してみましょう。プレイヤーの手を「チョキ」に変えてください。

```
21   // プレイヤーの手を決める
22   int player_hand = choki;
23
24   // コンピューターの手を決める
25   int cpu_hand = choki;
```

プログラムを書き換えたら「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。

「プレイヤーの手:グー」が表示されなくなっていたら成功です。

2.12 一定時間待つ

```
37   // 1秒待つ
38   usleep(1'000'000);
```

「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)。

こういう巨大な数値は桁数が分かりにくいものです。そのため、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)」を入れられるようになっています。

2.13 キーが押されるまで待つ

```
40  // 勝敗を表示する
41  printf("あなたの勝ちです！");
42
43  // なにかキーが押されるまで待つ
44  printf("なにかキーを押すと終了します");
45  wait_any_key();
```

「**wait_any_key**(ウェイト・エニー・キー)」を使うと、なにかキーが押されるまでプログラムの実行を停止できます。

3 手を表示しよう

3.1 チョキとパーを表示する

「チョキ」と表示されるようにするには、新しい**if**を追加します。33行目の下に、「プレイヤーの手がチョキだったとき、チョキと表示する」プログラムを書き加えてください。

```
27  // プレイヤーの手を表示する
28  if (player_hand == gu) {
29    printf("プレイヤーの手:グー");
30  }
31  if (player_hand == choki) {
32    printf("プレイヤーの手:チョキ");
33  }
34
35  // コンピューターの手を表示する
```

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。

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、選択した変換が決定します。入力が終わったらもう一度「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、日本語を入力するとき以外は常に「半角モード」を使います。

プログラムを書き換えたら「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。「プレイヤーの手:チョキ」と表示されたら成功です。

同様に、「パー」が表示されるようにしましょう。まず、プレイヤーの手を「パー」に変えてください。

```
21  // プレイヤーの手を決める
22  int player_hand = pa;
23
24  // コンピューターの手を決める
```

それから、33行目の下に、「プレイヤーの手がパーだったとき、パーと表示する」プログラムを書き加えてください。

```
31  if (player_hand == choki) {
32    printf("プレイヤーの手:チョキ");
33  }
34  if (player_hand == pa) {
35    printf("プレイヤーの手:パー");
36  }
37
38  // コンピューターの手を表示する
```

プログラムを書き換えたら「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。「プレイヤーの手:パー」と表示されたら成功です。

3.2 プレイヤーの手を画像で表示する

ここまで、「出した手」を文章で表示してきました。ゲームとしてはちょっと地味ですよね。そこで、「出した手」を画像で表示するように書き換えましょう。
28～29行目のprintfを、imageクラスのsetに書き換えてください。

```
27  // プレイヤーの手を表示する
28  image player_hand_image;
29  if (player_hand == gu) {
30    player_hand_image.set(200, 400, “gu.png”);
31  }
32  if (player_hand == choki) {
33    player_hand_image.set(200, 400, “choki.png”);
34  }
35  if (player_hand == pa) {
36    player_hand_image.set(200, 400, “pa.png”);
37  }
```

画像を表示するにはimageクラスを使うのでしたね。

プログラムを書き換えたら「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。

プレイヤーの手が、文章ではなく画像で表示されたら成功です。

3.3 コンピューターのグーとパーを表示する

現在、コンピューターの手を表示するプログラムは、「チョキ」だけが文章で表示されるようになっています(「コンピューター」だと文章が長くなって読みづらいので、ここからは代わりに「**CPU**(しーぴーゆー)」と書くことにします)。

プレイヤーの手と同様に、CPUの手を画像で表示するように書き換えてください。

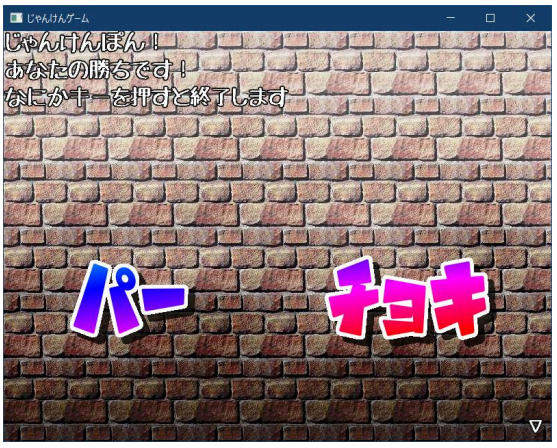
```
39  // コンピューターの手を表示する
40  image cpu_hand_image;
41  if (cpu_hand == choki) {
42    cpu_hand_image.set(600, 400, “choki.png”);
43  }
```

続いて、「グー」を表示するプログラムを書き加えてください。

```
39  // コンピューターの手を表示する
40  image cpu_hand_image;
41  if (cpu_hand == gu) {
42    cpu_hand_image.set(600, 400, "gu.png");
43  }
44  if (cpu_hand == choki) {
45    cpu_hand_image.set(600, 400, “choki.png”);
46  }
```

＜課題１＞
42行目の下に、CPUの「パー」を表示する**if**プログラムを書き加えなさい。

課題1が完了したら、「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。CPUの手が画像で表示されたら成功です。



4 勝敗を判定しよう

4.1 正しい勝敗を表示する

プレイヤーとコンピューターの手をいろいろと変えていますが、勝敗はいつも「あなたの勝ちです！」と表示されています。

原因は、勝敗を決めるプログラムを書いていないことです。そこで、勝敗を決めるプログラムを追加しましょう。

勝敗を決めるには「プレイヤーの手」と「CPUの手」の2つの条件を調べなくてはなりません。2つの条件を調べるには、**if**を二重に書きます。
54行目の勝敗を表示するプログラムに、次のプログラムを書き加えてください。

```
54  // 勝敗を表示する
55  if (player_hand == gu) {
56    if (cpu_hand == choki) {
57      printf("あなたの勝ちです！");
58    }
59  }
60
61  // なにかキーが押されるまで待つ
```

これで、「プレイヤーの手がグー、CPUの手がチョキ」の場合の正しい勝敗が判断できるようになりました。このプログラムは、

```
「プレイヤーの手がグーのとき、
   「コンピューターの手がチョキのとき、
        プレイヤーの勝ち」」
```

という構造になっています。ちょっとややこしいですね。

同じように、「プレイヤーの手がチョキ、CPUの手がパー」の場合の判定を作成しましょう。59行目の下に、次のプログラムを書き加えてください。

```
55  if (player_hand == gu) {
56    if (cpu_hand == choki) {
57      printf("あなたの勝ちです！");
58    }
59  }
60  if (player_hand == choki) {
61    if (cpu_hand == pa) {
62      printf("あなたの勝ちです！");
63    }
64  }
65
66  // なにかキーが押されるまで待つ
```

＜課題２＞
64行目の下に、プレイヤーの手が「パー」、CPUの手が「グー」の場合の勝敗を判定するifプログラムを書き加えなさい。

課題２が完了したら、「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。

「あなたの勝ちです！」が表示されなくなっていたら成功です。

4.2 コンピューターが勝利する条件を加える

これで、プレイヤーが勝つ条件を全て加えました。ですが、CPUが勝っているのに何も表示されないのはおかしいです。CPUが勝つ条件も書き加えましょう。

CPUが勝つ条件は、プレイヤーが勝つ条件と同じように、ifを書くことで判定できます。ですが、何度もifを書くのは面倒ですよね。なんとか、ひとつのifで判定できないものでしょうか。

プレイヤーの手が「パー」、CPUの手が「チョキ」の場合を考えます。
paは2、chokiは1なので、「プレイヤーの手 - CPUの手」の計算結果は1です。
プレイヤーの手が「チョキ」、CPUの手が「グー」の場合は「1 - 0」で、これも差は1です。もしかして、引き算の差を見るだけで勝ち負けが分かるかも…?

ところが、プレイヤーの手が「グー」、CPUの手が「パー」のとき、差は「-2」になってしまいます。

プレイヤーの手	CPUの手	差
グー(0)	パー(2)	-2
チョキ(1)	グー(0)	1
パー(2)	チョキ(1)	1

どうも、うまくないですね。しかし、あきらめるには早いです。何か工夫をすれば、勝ちを判定できるかも。3を足して、「グー」対「パー」の差を無理やり1にします。

プレイヤーの手	CPUの手	差
グー(0+3)	パー(2)	1
チョキ(1)	グー(0)	1

パー(2)	チョキ(1)	1

いちおう、全部1になりましたね。ですが、これだと「グーだったら3を足す」という判定を追加しないといけません。判定の数を減らしたいのに、新しく判定を追加するのでは本末転倒です。ということで、チョキとパーにも3を足してみます。

プレイヤーの手	CPUの手	差
グー(0+3)	パー(2)	1
チョキ(1+3)	グー(0)	4
パー(2+3)	チョキ(1)	4

…うまくいきませんね。やっぱり無理なんでしょうか…。あ、いえ、ちょっと待ってください。3で割ったらどうでしょう。すると、あまりは以下の表のようになります。

プレイヤーの手	CPUの手	差	差を３で割ったあまり
グー(0+3)	パー(2)	1	1
チョキ(1+3)	グー(0)	4	1
パー(2+3)	チョキ(1)	4	1

やった！　全部1になりましたね。まとめると、次のルールが成り立ちそうです。

「((プレイヤーの手+3)-CPUの手)÷3」を計算し、あまりが「１」ならCPUの勝ち

それでは、このルールを使ってCPUの勝ち判定を書きましょう。
C++言語で「あまり」を計算するときは「**%**(パーセント)」記号を使います。52行目の勝敗を表示するプログラムに、CPUが勝つ条件の**if**を書き加えてください。

```
54   // 勝敗を表示する
55   int amari = ((player_hand + 3) - cpu_hand) % 3;
56   if (amari == 1) {
57     printf("あなたの負けです");
58   }
59
60   if (player_hand == gu) {
```

書きかえたら「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。「あなたの負けです。」と表示されたら成功です。

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。

4.3 「あいこ」の条件を加える

「あいこ」の条件も付け加えましょう。
あいこになるように、CPUの手を書きかえてください。

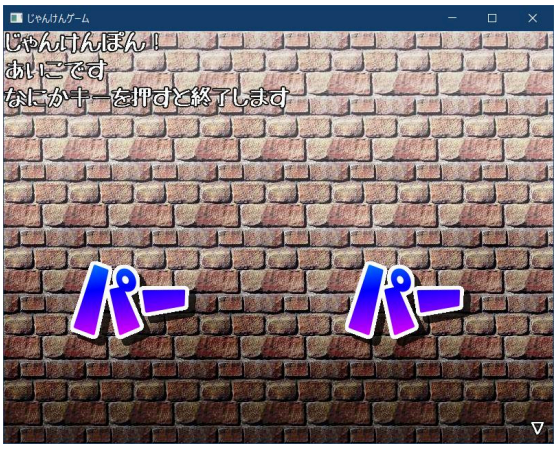
```
24|   // コンピューターの手を決める
25|   int cpu_hand = pa;
```

あいこの判定には、さきほどCPUの勝ち判定のために計算した「あまり」が使えます。なんと、引き分けのときは「あまり」が「0」になるのです！

40行目のあまりを求めるプログラムの下に、あいこ用のifを書き加えてください。

```
54|   // 勝敗を表示する
55|   int amari = ((player_hand + 3) - cpu_hand) % 3;
56|   if (amari == 0) {
57|     printf("あいこです");
58|   }
59|   if (amari == 1) {
60|     printf("あなたの負けです");
61|   }
```

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。「あいこです」と表示されたら成功です。



この章で見たように、プログラムの書き方を工夫すると、プログラムが見やすく、分かりやすくなります。

なぜ「3で割った余り」で勝敗が分かるのか？

「整数ｎがあるとする。0+3nは1+3nに勝ち、1+3nは2+3nに勝ち、2+3nは0+3nに勝つ。」というルールを考えます。このルールの下では、手にどれだけ3を足し引きしても、勝敗は変わりません。これを「n=整数じゃんけん」と名付けましょう。
実は、「3で割ったあまり」で判定する方法は、この「n=整数じゃんけん」のルールを使ったものなんです。
この「n=整数じゃんけん」をn=0の場合に限定すると、ふつうのじゃんけんになります。つまり「n=0じゃんけん」というわけです。
当然、「n=0じゃんけん」と「n=整数じゃんけん」のルールは同じです。だから、どちらのルールを使っても、正しい勝敗が判定できるわけです。

5 手を選べるようにしよう

5.1 プレイヤーの手を選ぶ

プログラムに手を書いているので、出す手を変えるにはプログラムを書き換えなくてはなりません。

これは面倒なので、実行するときに手を選べるようにしましょう。
プレイヤーの手を決めるプログラムを、次のように書き換えてください。

```
21|   // プレイヤーの手を決める
22|   int player_hand = select(3, "グー", "チョキ", "パー");
23|
24|   // コンピューターの手を決める
25|   int cpu_hand = choki;
```

「**select**(セレクト)」を使うと、選択肢を表示してプレイヤーに選んでもらうことができます。
最初の数字が選択肢の数、そのあとの3つが選択肢として表示される文章です。

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。グー、チョキ、パーの選択肢が表示されたら成功です。

上キーと下キーで好きな手を選び、**Enter**キーを押すと決定されます。

5.2 コンピューターの手を選ぶ

今度はコンピューターの手が毎回変化するようにしましょう。

```
21|   // プレイヤーの手を決める
22|   int player_hand = select(3, "グー", "チョキ", "パー");
23|
24|   // コンピューターの手を決める
25|   int cpu_hand = rand() % 3;
```

「rand(ランド)」は正の整数を無作為に選んで返します。

ただし、返される数値の範囲はとても広い(Visual Studioの場合0〜32767)です。そのままでは使いにくいので、なんらかの計算を行って、必要な数値に変換します。

例えば、ある範囲の数値が欲しい場合は「あまり」を計算します。

今回、必要なのは、0〜2の数値です。そこで、「3で割ったあまり」を計算します。こうして選ばれた0、1、2のいずれかの数値が、コンピューターの手になります。

書きかえたら「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、勝敗が変化したら成功です。

6 音声を鳴らそう

6.1 効果音を鳴らそう

じゃんけんする時に小鼓(こつづみ)の音を鳴らしてみましょう。
音声を鳴らすには「play_sound(プレイ・サウンド)」を使います。プレイヤーとCPUの手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。

```
47   if (cpu_hand == pa) {
48     cpu_hand_image.set(600, 400, "pa.png");
49   }
50
51   play_sound("kotudumi.mp3");
52
53   // 1秒待つ
54   usleep(1'000'000);
```

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。音量は右下のスピーカーアイコンから変更できます。

20〜30くらいがいいと思います。大丈夫そうならボタンをクリックして実行してください。選択肢を決定したとき、小鼓の(ような)音声が再生されたら成功です。

6.2 BGMを鳴らそう

じゃんけんを始める前に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。
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("じゃんけんゲーム");
 8
 9   play_bgm("bgm_normal.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３＞
プレイヤーが負けたときにse_lose.mp3という効果音を再生しなさい。
音声ファイルはResというフォルダに入っています。

7 何度でも遊べるようにしよう

7.1 プログラムを繰り返し実行するには

一度プログラムを実行したら、終了せずに何度でも遊べ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。

じゃんけんゲームのほぼ全体が繰り返されるようにしたいので、play_bgmのすぐ下に「**for (;;) {**」を書き加えてください。

```
 9|   play_bgm("bgm_normal.mp3");
10|
11|   for (;;) { // 繰り返しの開始点
12|     // 背景を表示
13|     image background;
```

それから、**wait_any_key**と**finalize**の間に「**}**(閉じカッコ)」を書き加えてください。

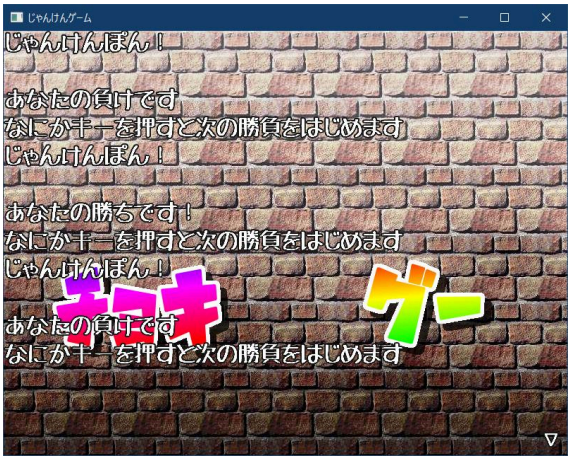
```
84|     // なにかキーが押されるまで待つ
85|     printf("なにかキーを押すと終了します");
86|     wait_any_key();
87|   } // 繰り返しの終了点
88|
89|   // プログラムの終了処理
90|   finalize();
```

これで何度もじゃんけんを楽しめるようになりました。

そうなると、「なにかキーを押すと終了します」という文章はおかしいですね。
「なにかキーを押すと次の勝負をはじめます」という文章に書きかえてください。

```
84      // なにかキーが押されるまで待つ
85      printf("なにかキーを押すと次の勝負をはじめます");
86      wait_any_key();
87    } // 繰り返しの終了点
```

書きかえたら「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。何度でもじゃんけんができるようになっていれば成功です。



7.2 画像と文字を消そう

じゃんけんを続けると、前のじゃんけんのときの文字や画像が表示されたままになっていることに気づくと思います。見づらいので、次の勝負が始まる前に、これらの文字や画像を消しましょう。

文章を消すには「clear_all_text(クリア・オール・テキスト)」を使います。
画像を消すには「clear_all_image(クリア・オール・イメージ)」を使います。

wait_any_keyと「}」のあいだに、次のプログラムを書き加えてください。

```
74      // 何かキーが押されるまで待つ
75      printf("何かキーを押すと次の勝負をはじめます");
76      wait_any_key();
77
78      // 次の勝負に備えて文章と画像を消す
79      clear_all_text();
80      clear_all_image();
81    } // 繰り返しの終了点
82
83    // プログラムの終了処理
84    finalize();
```

プログラムを書き加えたら実行してください。じゃんけんを続けたとき、前に表示されていた文章と絵が消えていたら成功です。

8 完成！

おめでとうございます！

これでじゃんけんゲームのテキストは終了です。最後に、いくつかの練習問題を用意しました。それが終わったら、あとはあなたの好きなように改造していってください。

お疲れ様でした。

＜練習１＞
プレイヤーが勝ったときに**se_win.mp3**という効果音を再生しましょう。

＜練習２＞
プレイヤーの勝利判定を行う**if**を、**amari**を使う方法で書き直してみましょう。

＜練習３＞
対戦相手を画像で表示してみましょう。
画像ファイルは**Res**という名前のフォルダに入っています。

＜練習４＞
勝敗を文章ではなく画像で表示してみましょう。
Resフォルダから使えそうな画像を探してみましょう。

＜練習５＞
あなたの好きなようにゲームを改造してください。
Resフォルダの画像や音声は自由に使って構いません。

https://github.com/tn-mai/Rock-Paper-Scissors