[バンタン体験授業(2019年11月版)]

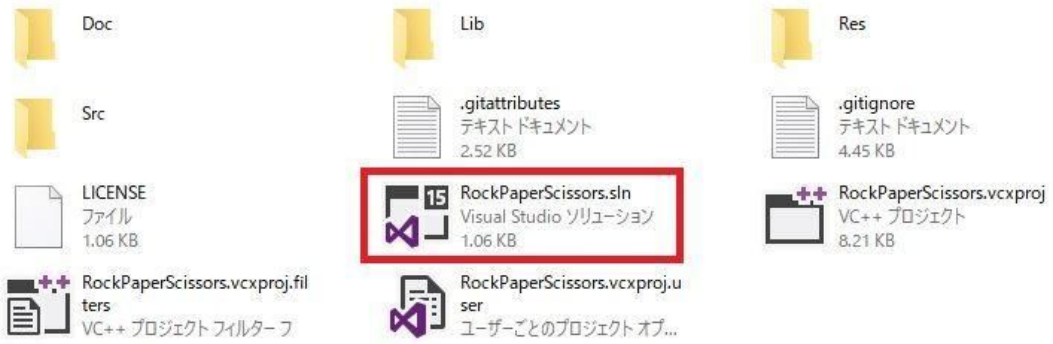
じゃんけんゲームを作ってみよう

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「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。

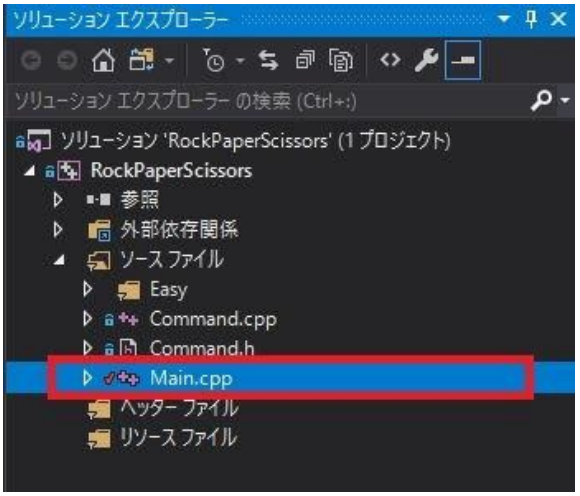
よく分かりませんが、とりあえず指示に従って何かキーボードのキーを押してみましょう。そうすると、ウィンドウが閉じると思います。

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、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。

それでは、Main.cppを見ていきましょう。しかし、その前にひとつ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んでください。

- 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。

2.3 インクルード指令

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

「#include」(シャープ・インクルード)は「インクルード指令」といって、プログラムで使う部品を選択します。「Command.h」(コマンド・ドット・エイチ)というのが部品ファイルの名前です。Command.hには、この「じゃんけんゲーム」で使われているさまざまな部品が含まれています。

2.4 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 {
```

プログラムは「int main」(イント・メイン)で始まります。mainのすぐ後ろには「(」(丸かっこ)と「)」(閉じ丸かっこ)があり、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。コンピューターはコメントを無視します。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。

2.5 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。丸かっこと閉じ丸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。C++では、「"」(ダブル・クォーテーション)という記号で囲むことで、この範囲はプログラムではなくて文章ですよ、ということを示します。

```
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。

2.6 画像を表示する

```
 9  // 背景を表示
10  set_image(No_0, 400, 300, "bg_paper.jpg");
```

「set_image」(セット・イメージ)は画面に画像を表示します。丸括弧の内側には「,(カンマ)」で区切られた4つのパラメーターがあります。

最初の「No_0」は画像の管理番号です。これは、コンピューターが表示する画像を管理するための番号です。管理番号は同時に表示する画像ごとに違う番号を指定します。その次の「400」と「300」は画像を表示する位置で、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。ウィンドウの大きさは横が800、縦が600です。つまり、ウィンドウの中心に画像を表示しようとしているのですね。

最後にある「"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。

ゲームの雰囲気を変えるために、背景を変えてみましょう。10行目のファイル名を、次のように書き換えてください。

```
 9  // 背景を表示
10  set_image(No_0, 400, 300, "bg_brick.jpg");
```

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。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。

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。

2.7 文章を表示する

```
12|   // 文章を表示
13|   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。

2.8 数値に名前を付ける

```
15|   // じゃんけんの手を定義する
16|   int gu = 0;
17|   int choki = 1;
18|   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ですが、文字の並びを扱うのはあまり得意ではありません。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「数値に名前を付ける機能」が用意されています。数値に名前をつけるには

```
int 名前 = 数値;
```

と書きます。この式は「これから整数に名前を付けますよ」という意味です。「int」(イント)は「integer(インテジャー、「整数」という意味)」を省略したもので、「名前 = 数値」の部分で数値に名前を付けています。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。

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。

［名前が使える範囲］
名前は、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。

2.9 数値に名前をつける(その2)

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

2.10 一定時間待つ

```
26|   // 1秒待つ
27|   wait(1);
```

「wait」(ウェイト)を使うと、指定した秒数だけプログラムの実行を停止できます。

2.11 条件によって違うことをする

```
29|   // 勝敗を判定する
30|   if (player_hand == gu && cpu_hand == choki) {
31|     printf("あなたの勝ちです！");
32|   }
```

条件に応じて違うことをするには「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。

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。2つ以上の条件を調べるには、条件のあいだに「&&」(アンド・アンド)記号を書きます。つまり、このifの条件は、

プレイヤーの手がグーのとき	かつ	コンピューターの手がチョキのとき
player_hand == gu	&&	cpu_hand == choki

という意味になります。

ためしに、すこし出す手を変えてみましょう。
プレイヤーの手とコンピューターの手を、次のように書き換えてください。

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

手を書き換えたら「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。

…おや？
勝敗が表示されなくなってしまいましたね。

勝敗が表示されなくなった理由は、30行目のifに書かれた条件にあります。現在の条件は「プレイヤーの手がグー、かつ、コンピューターの手がチョキの場合」なので、それ以外の手を指定しても、条件を満たせないのです。

他の条件でも勝敗を表示するには、ifを増やして対応できる条件を増やすのが簡単です。次のように30～32行目にプログラムを書き加えてください。

```
29  // 勝敗を判定する
30  if (player_hand == gu && cpu_hand == choki) {
31    printf("あなたの勝ちです！");
32  }
33  if (player_hand == choki && cpu_hand == pa) {
34    printf("あなたの勝ちです！");
35  }
36
37  // 何かキーが押されるまで待つ
```

書き加えたら「ローカルWindowsデバッガー」をクリックして実行しましょう。

「あなたの勝ちです！」と表示されたら成功です。

3 勝敗条件を追加しよう

3.1 パーVSグーの条件を加える

グーVSチョキ、チョキVSパーの次は、パーVSグーのifを書き加えましょう。
まずプレイヤーとコンピューターの手を書き換えてください。

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

そして、パーVSグーのifを書き加えてください。

```
33  if (player_hand == choki && cpu_hand == pa) {
34    printf("あなたの勝ちです！");
35  }
36  if (player_hand == pa && cpu_hand == gu) {
37    printf("あなたの勝ちです！");
38  }
39
40  // 何かキーが押されるまで待つ
```

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。
「あなたの勝ちです！」と表示されたら成功です。

3.2 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。しかし、このままではコンピューターが勝ったとしても何も表示されません。ちょっと可哀想なので、コンピューターが勝つ条件も加えましょう(「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)。

プレイヤーが勝利する条件と同じように、３つのifを書くことで判定できます。でも、また３回も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、あきらめるには早すぎます。なにか工夫をすれば、勝ちを判定できるかも。「グー」対「パー」の差を無理やり1にするために、とりあえずプレイヤーの手に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)	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。これを、3で割ってみたらどうでしょう。そうすると、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！

ここまでをまとめると、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」を計算して、あまりが「1」ならCPUの勝ち

それでは、このルールを使ってコンピューターの勝ち判定を書きましょう。
まず、プレイヤーとコンピューターの手を、以下のように書き換えてください。

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

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。プレイヤーが勝つ条件のifの下に、コンピューターが勝つ条件のifを書き加えてください。

```
36  if (player_hand == pa && cpu_hand == gu) {
37    printf("あなたの勝ちです！");
38  }
39  int amari = ((player_hand + 3) - cpu_hand) % 3;
40  if (amari == 1) {
41    printf("あなたの負けです");
42  }
43
44  // 何かキーが押されるまで待つ
```

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。
理由は、古いコンピューターでは掛け算や割り算の記号が使えなかったからです。そこで、当時のプログラミング言語の開発者たちは、掛け算には「＊」、割り算には「/」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。

3.3 「あいこ」の条件を加える

じゃんけんでは、双方が同じ手を出した場合は「あいこ」になります。
だから、あいこの条件も付け加えましょう。まずはあいこになるように手を変えてください。

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

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！　納得できないようでしたら、先に出てきたルールに数字を当てはめて計算してみてください。納得できたら、次のようにあいこのifを書き加えてください。

```
40  if (amari == 1) {
41    printf("あなたの負けです");
42  }
43  if (amari == 0) {
44    printf("あいこです");
45  }
46
47  // 何かキーが押されるまで待つ
```

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、プログラムの見やすさ、分かりやすさは大きく変化します。

4 手を選べるようにしよう

4.1 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。

```
20|   // プレイヤーの手を決める
21|   int player_hand = select(3, "グー", "チョキ", "パー");
22|
23|   // コンピューターの手を決める
24|   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ます。Enterキーを押すと決定されます。

4.2 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。

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

「random」(ランダム)は正の整数を無作為に選んで返します。
返される数値の範囲はとても大きいので、ある範囲の数値が欲しい場合は「あまり」の計算を使います。今回ほしいのは0～2の数値なので、3で割った「あまり」を計算すればよいわけです(「あまり」を求めるには「%」(パーセント)記号を使うのでしたね)。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毎回同じ手を選んでください。勝敗が変化したら成功です。

5 手を表示しよう

5.1 プレイヤーの手を表示しよう

じゃんけんの結果は表示されますが、実際に何を出したのかは分かりません。
インチキされてるのかもしれませんよね。そこで、出した手を表示することにしましょう。
プレイヤーの手を決め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。

```
20  // プレイヤーの手を決める
21  int player_hand = select(3, "グー", "チョキ", "パー");
22
23  xyprintf(100, 260, "[あなたの手]");
24  if (player_hand == gu) {
25    set_image(No_1, 200, 400, "gu.png");
26  }
27
28  // コンピューターの手を決める
29  int cpu_hand = random() % 3;
```

23行目のxyprintf(エックス・ワイ・プリント・エフ)は、最初の2つの数値で表示位置を自由に決められるバージョンです(注: xyprintfはこのテキストのために作成したもので、一般のC++言語には存在しません)。

set_imageの管理番号は「No_1」にしています。No_0は背景を管理するために使っているので、別の番号を指定しなければならないからです。この体験プログラムでは、No_0からNo_19まで全部で20個の管理番号が使えます。現在はNo_0しか使っていないので、No_1以降はどれでも使えます。しかし、あまり適当に番号を付けると、あとで何番を使っているのかが分からなくなってしまいますので、順番に割り当てることにしています。

グーの画像に変わるはずです。No_0は背景の管理番号だからです。

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。グーを選んだとき、グーの画像が表示されたら成功です。

＜練習１＞
グーの画像を表示するプログラムの下に、チョキとパーの画像を表示するifを書き加えましょう。
チョキの画像は”choki.png”、パーの画像は”pa.png”を使ってください。

5.2 コンピューターの手を表示しよう

プレイヤーの手と同じやりかたで、コンピューターの手も表示しましょう。コンピューターの手を決め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。

```
34  // コンピューターの手を決める
35  int cpu_hand = random() % 3;
36
37  xyprintf(450, 260, "[コンピューターの手]");
38  if (cpu_hand == gu) {
39    set_image(No_2, 600, 400, "gu.png");
40  }
41
42  // 1秒待つ
43  wait(1);
```

コンピューターの手の文章や画像は、プレイヤーのそれと重ならないように座標を調節してあります。また、背景やプレイヤーの手の画像を変えてしまわないために、管理番号にはNo_2を使っています。

＜練習２＞
コンピューターのグーの画像を表示するプログラムの下に、チョキとパーの画像を表示するifを書き加えましょう。

＜練習２＞ができたら、「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。コンピューターの手に対応する画像が表示されたら成功です。

5.3 音声を鳴らそう

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。

```
44  if (cpu_hand == pa) {
45    set_image(No_2, 600, 400, "パー.png");
46  }
47
48  play_sound("kotudumi.mp3");
49
50  // 1秒待つ
51  wait(1);
```

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。

5.4 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  set_image(No_0, 400, 300, "bg_brick.jpg");
```

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。

5.5 何度でも遊べるようにしよう

一度実行すれば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。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、何度でも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、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 (;;) {
12    // 背景を表示
13    set_image(No_0, 400, 300, "bg_brick.jpg");
```

それから、「何かキーを押すと終了します」という文章を「何かキーを押すと次の勝負をはじめます」というふうに書きかえ、さらにwait_any_keyとfinalizeの間に「}」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
79  // プログラムの終了処理
80  finalize();
```

書き加えたら実行してください。
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。

5.6 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「reset_all_text」(リセット・オール・テキスト)を使います。
画像を消すには「reset_all_image」(リセット・オール・イメージ)を使います。

それでは、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      // 次の勝負に備えて文章と画像を消す
78      reset_all_text();
79      reset_all_image();
80    }
81
82    // プログラムの終了処理
83    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。

6 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。
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３＞
勝敗によって異なる音声を再生してみましょう。
音声はResというフォルダに入っています。

＜練習４＞
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５＞
勝敗を文章ではなく画像で表示してみましょう。
画像は「Res」という名前のフォルダに入っています。

＜練習６＞
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。