Introduction to Programming

はじめてのプログラミング　C言語

明治大学
教育の情報化推進本部
（和泉）

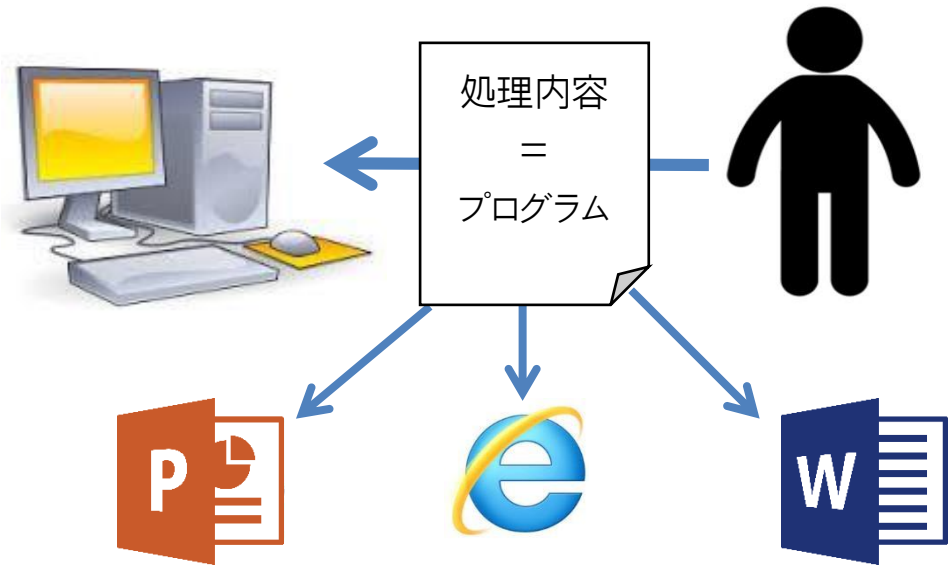
201904

1. はじめに

「プログラミング」という言葉を聞いたことがあるでしょうか？初めて聞いた人もいるかもしれませんが、皆さんは普段の生活の中で知らないうちにプログラミングの恩恵を受けています。例えば、スマートフォンを利用する際に、様々なアプリを利用していると思います。このアプリ、アプリケーションはプログラムによって構成されています。スマートフォンに限らず電車のダイヤ、銀行のシステム、テレビ、車の制御など、現代の人々の生活はプログラミングによって支えられているといっても過言ではありません。この講座を通して、「プログラムって何？」、「プログラミングって何？」、「C 言語って何？」という概念的な理解をし、最後には「プログラミングをすることができた」と実感していただけるようになれば幸いです。

2. 「プログラム（アプリ）」と「プログラミング」の違い

「プログラム ＝ 人間がコンピュータにさせる処理の内容」と言えます。
コンピュータは人間の様に自律的に動きません。コンピュータに何らかの処理をさせるためには、その内容を人間が教えてあげなければなりません。例えば、皆さんが Web サイトを閲覧する時に使う Internet Explorer はプログラムです。「インターネットに接続して Web サイトを表示しろ」という処理内容が Internet Explorer というプログラムと言えます（厳密にはこんな簡単な内容ではありませんが、かなり広い意味で説明させていただきます）。

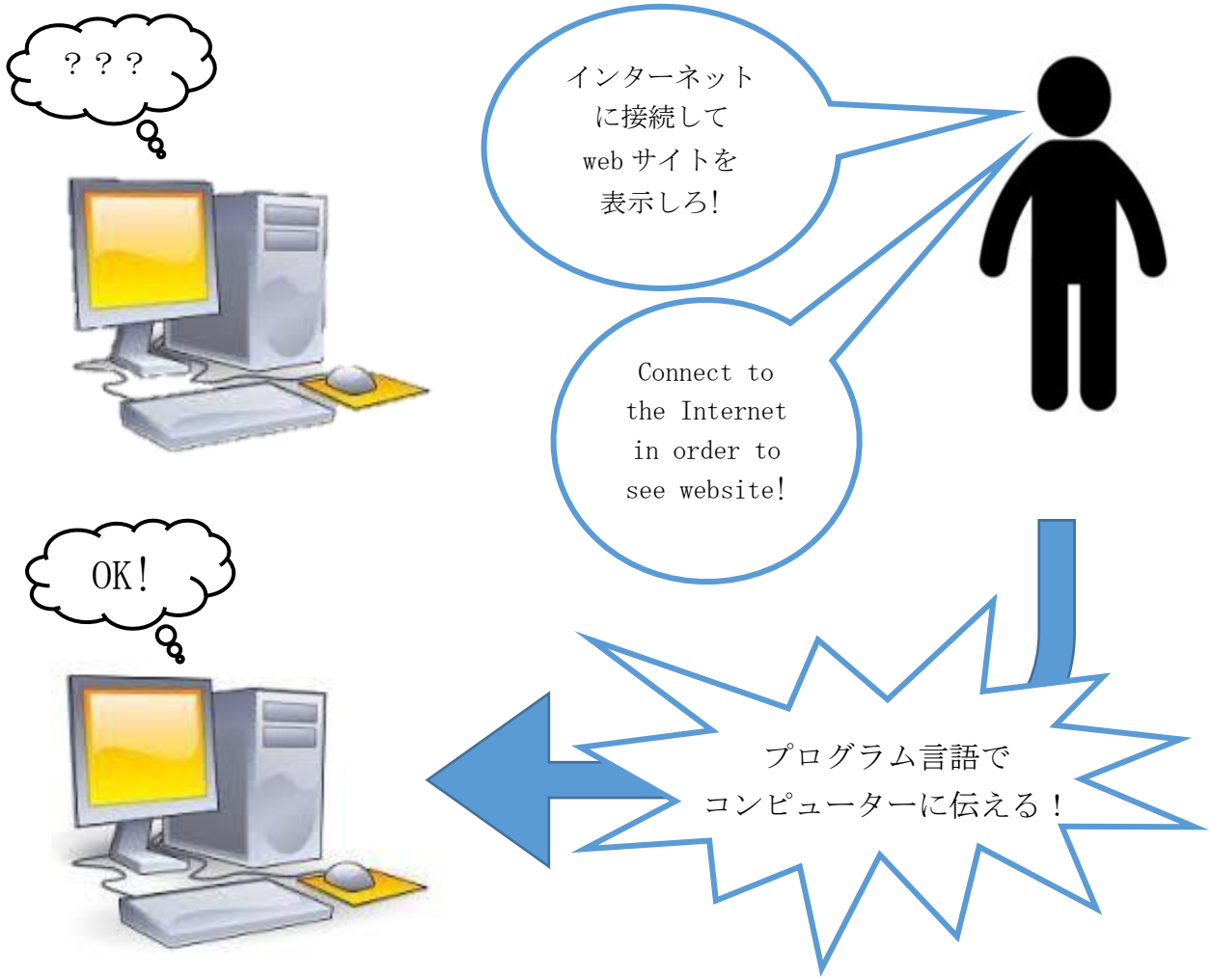


「プログラミング ＝ プログラムを作るための行為」を指します。
プログラミングを職業にしている人をプログラマーと呼びます。

3. C 言語って何？

前章において、「プログラム＝人間がコンピュータにさせる処理の内容」と学びました。では、人間がコンピュータにさせる処理内容をどうやって表現したらいいでしょうか？コンピュータは日本語を理解してくれるでしょうか？英語を理解してくれるでしょうか？仮に日本語や英語を理解してくれるとしても、人間は文章の作り方が一様ではありません。例えば、「インターネットに接続して、Web サイトを表示しろ」とコンピュータに伝えることを考える場合、「Web サイトが見たいから、インターネットに接続しろ」もほぼ同じ意味になりますよね？前にも説明しましたが、コンピュータは人間の様に自律的に動きません。そのため、「人間とコンピュータが相互に理解しあえる言葉」を定義する必要があります。その言葉の 1 つが今回学ぶ C 言語です。他にも様々な言葉（Java、PHP、Perl、Ruby、Python、Java script、etc）がありますが、これらの様々な言語の多くは、C 言語から派生した方言のようなものです。そのため、C 言語を理解すれば、その他の言語はだいたい扱うことができます。

もちろん最初から難しいことはできませんが本日はプログラミングの第１歩を踏み出してみましょう。



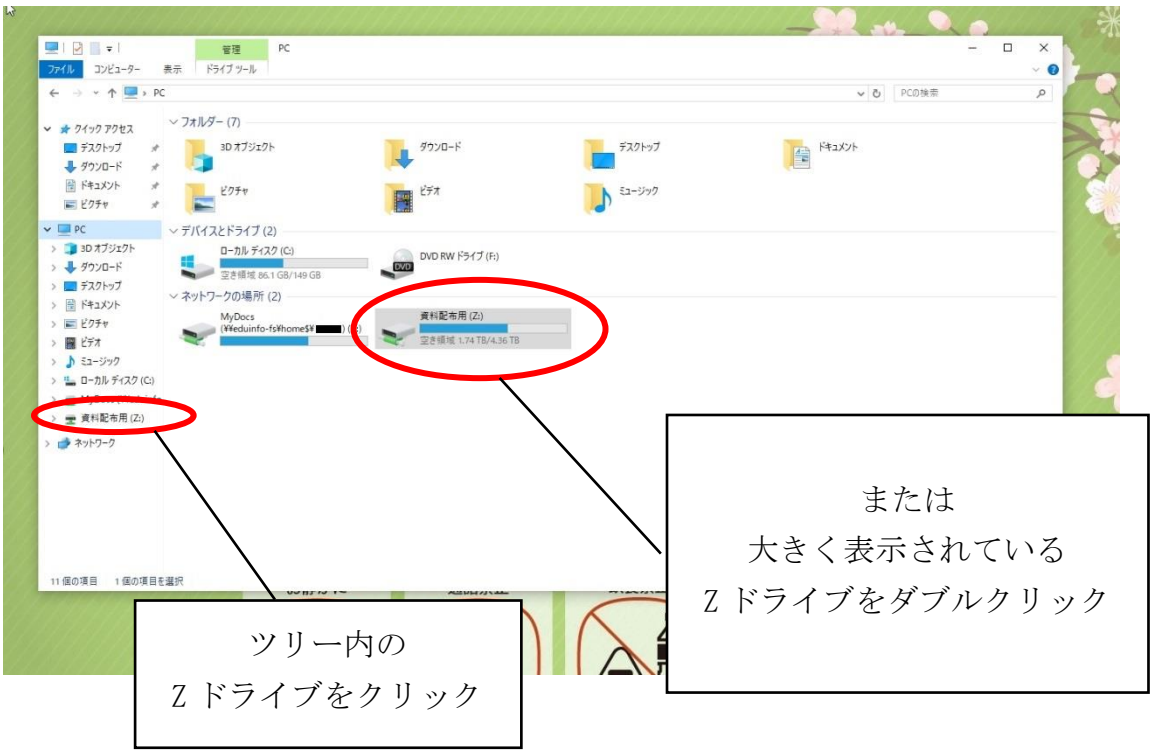
4. C 言語でプログラムを書いてみよう（＝プログラミングをしよう）

事前準備

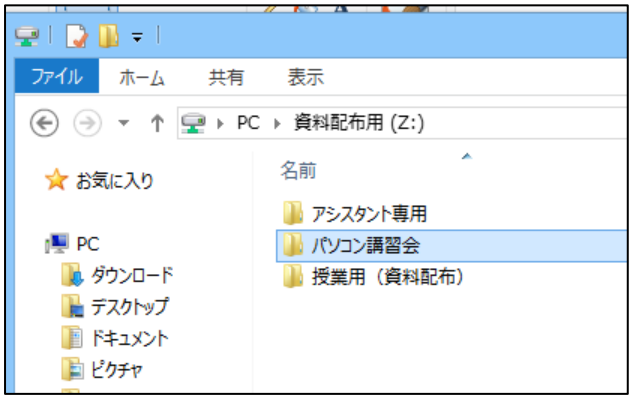
以下の手順に従って、プログラミングをするための準備をしましょう。

画面左下にある「エクスプローラー」(下図参照)をクリック
（フォルダマークをクリック）

エクスプローラーが起動しますので 資料配布用 Z ドライブ を開く



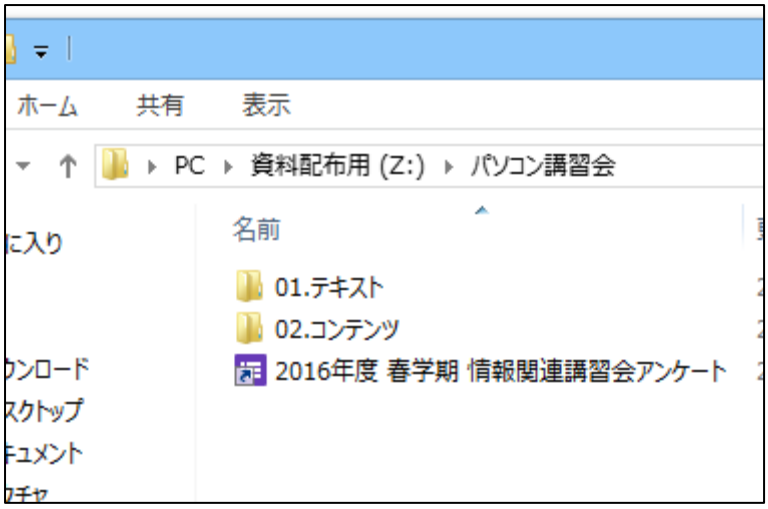
以下の画面が表示されるので、**「パソコン講習会」**をひらく



さらに**「02.コンテンツ」**をひらく

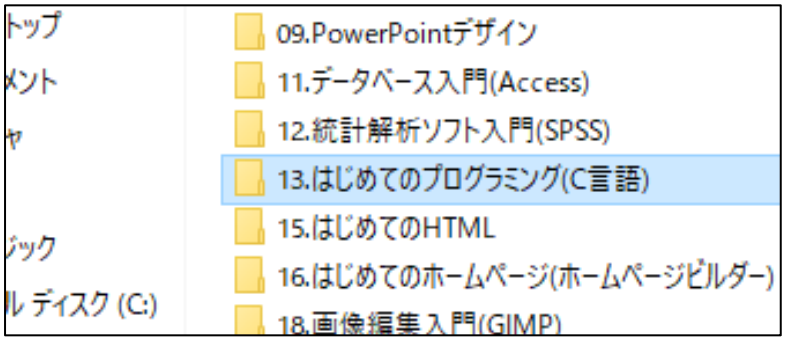
※「01.テキスト」に全講習会のテキストが入っています。

興味がある方は見てみましょう。



※〇〇年度　〇学期　情報関連講習会アンケートは講習会の最後に行います。
　ご協力お願いします。

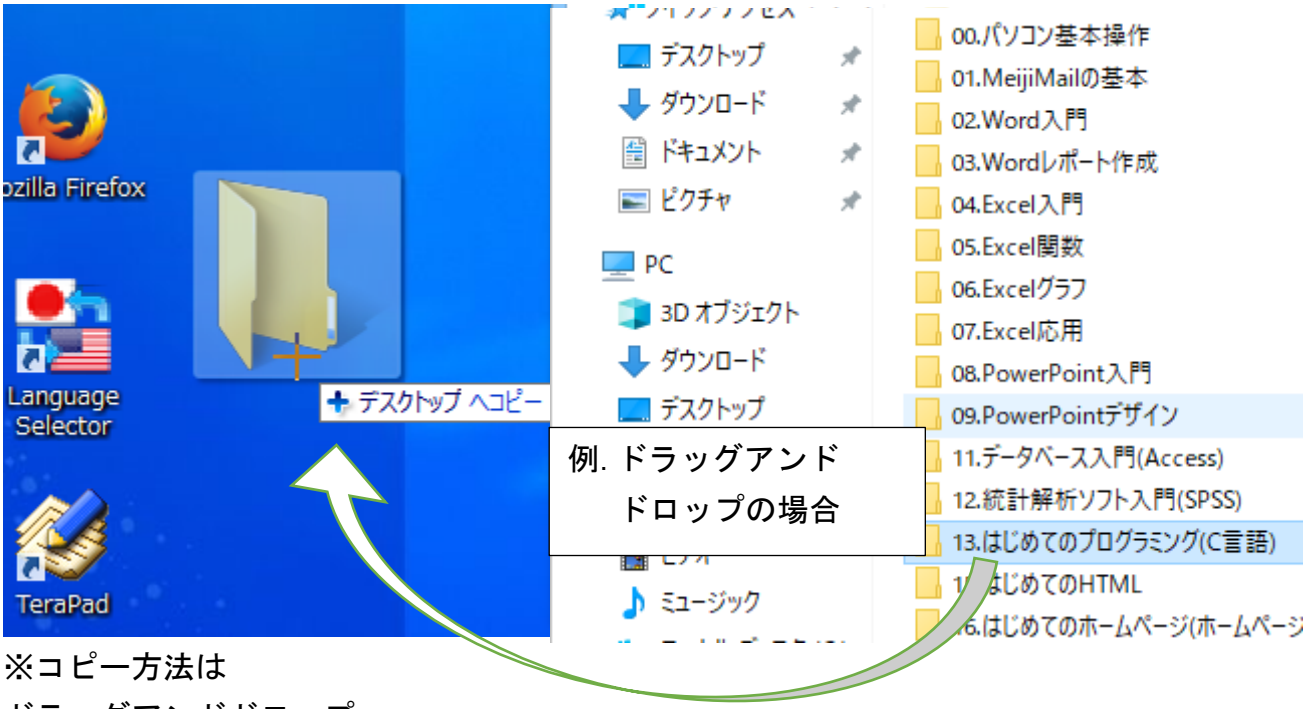
コンテンツ内の**「 13.はじめてのプログラミング(C言語)」**を選択します。



13.はじめてのプログラミング（C言語）をフォルダごと

MyDocs(X)またはデスクトップにコピーしてください。

※デスクトップに保存した場合はログオフ、再起動後、消去されます。



※コピー方法は

ドラッグアンドドロップ

右クリックでコピーを選んで保存する場所で右クリック貼り付け

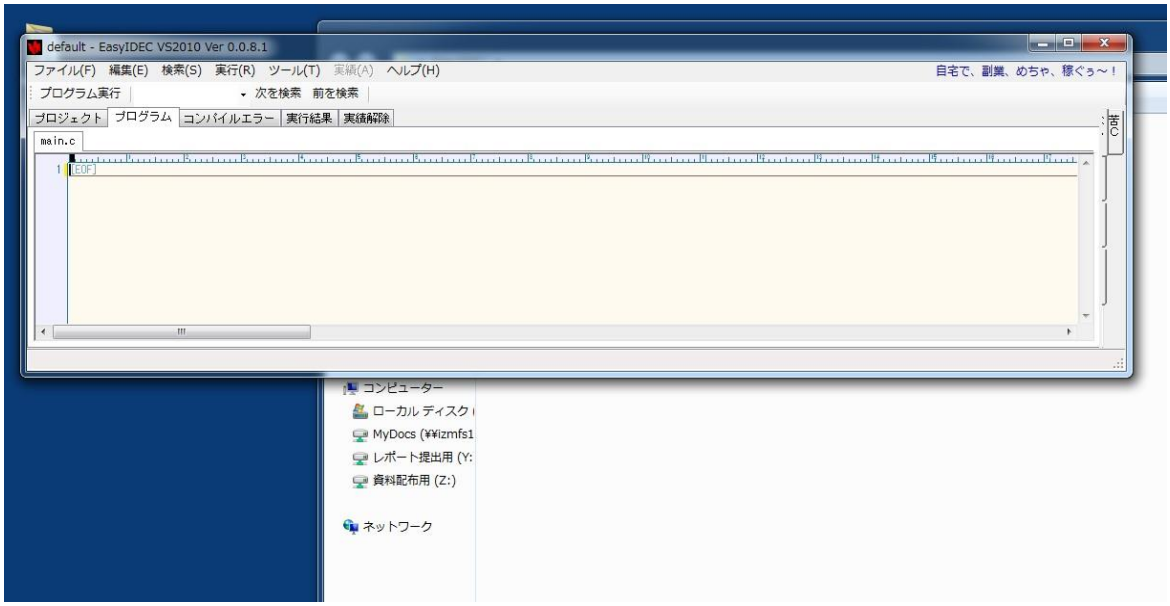
（キーボードショートカットでやるなら　ctrl+C　からの保存場所で ctrl+V）

などです。

コピーで作成したフォルダ内の「easyIDEC」を開き、「EasyIDEC.exe」を起動します(ダブルクリックして下さい)。以下の画面が表示されれば、準備完了です。

※最初に出る「スクリプトエラー」のウィンドウはすべて「はい」をクリックします。
何度も出ますが、すべて消してください。

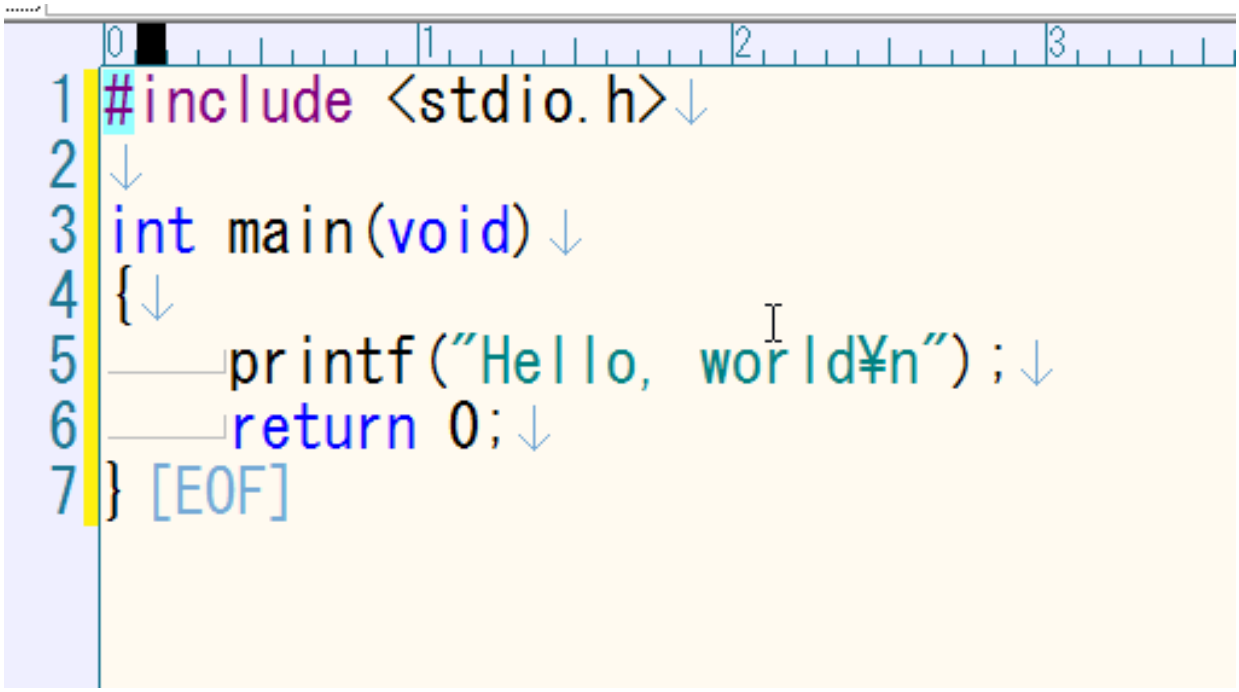
ウィンドウを見やすい大きさに調整しましょう。



さぁプログラムをしましょう！

以下の通りに文章を入力してみましょう。

入力する文字は**全て半角英数字**です。



入力方法

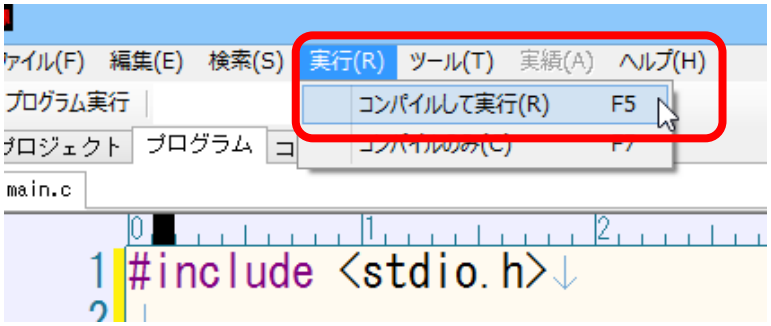
＝ shift + 3 ， ＜ ＝ shift + , ， ＞ はその隣

{ ＝ shift + [[はエンターキーの左横， ″ ＝ shift + 2

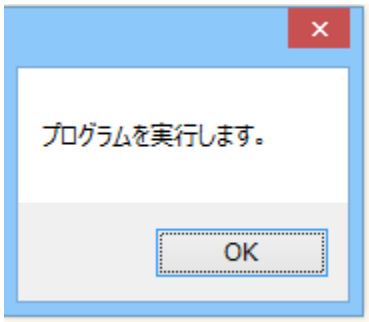
¥ は Backspace の左横

このプログラムは、「「Hello, world」 と画面に表示しろ 」という処理内容を C 言語で記述したものです。

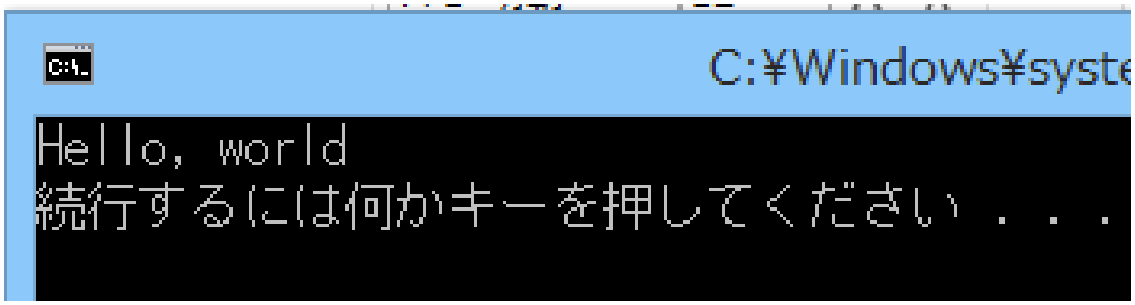
書き終えたら上のメニュー「実行」から「**コンパイルして実行**」を選択してください。



※ **コンパイルして実行**は Ｆ５ キーがショートカットになります。
この後よく使うので Ｆ５ を覚えてしましましょう。



正しくプログラムがかけている場合は「プログラムを実行します」というウィンドウが出て「ＯＫ」をクリックすると下のような画面が出ます。



上のような画面ができた人は以下をやってみましょう。どうなるでしょう

1. 「Hello, world」の部分を自分の名前など適当な単語（日本語全角）にする

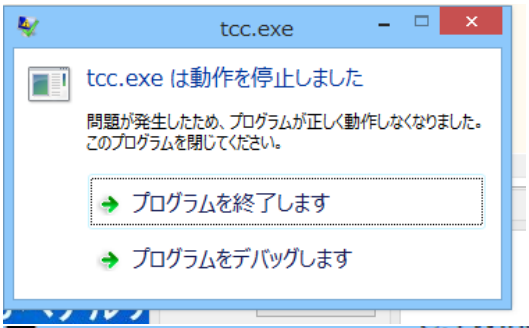
2.「　¥n　」を消す。

3.「　¥n　」の後ろにも文字を追加する。
　例「Hello,　wolrd¥n 明治大学」「Hello,　wolrd ¥n Izumi campus」など

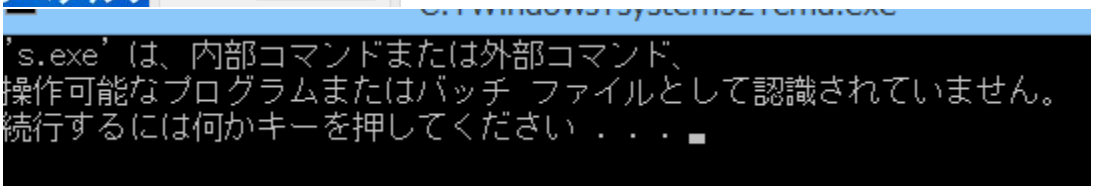
4.「　¥n　」を増やしてみる
　例 Hello の一文字ずつに¥n を挟む「　H¥ne¥nl¥nl¥no¥n　」

実行したらエラーが出た、という人は次のページに進みましょう。

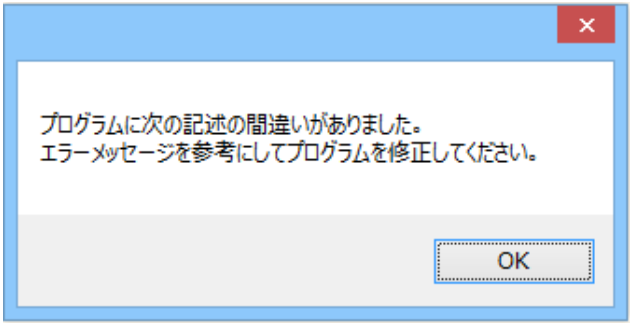
エラー



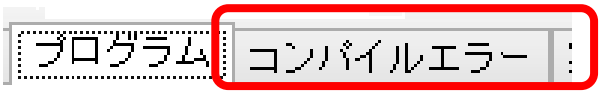
左のようなエラーが出た場合
コピーしたフォルダ内の「Easy IDEC.exe」
ではなく、コピー元Ｚドライブの「Easy IDEC.exe」を起動しています。
コピーしたフォルダの中にある「Easy IDEC.exe」
を起動しなおしてください。



もしコンパイルができたうえでプログラムに何か間違っている箇所がある場合は左図のような「・・・間違いがありました。・・・」というメッセージが出ます。



エラーメッセージを消し
（「ＯＫ」をクリックし）
どこが間違っているかを確認しましょう。



タブ「コンパイルエラー」にエラー内容が指摘されます。上から１つずつ指摘されるのでエラーがたくさんある場合は、修正→修正→修正→・・・・と続けていきます。

「3行目」で記述エラーを発見しました。
「;」を付け忘れています。

「；」を付け忘れています。と記述されている場合

1. 実際に「；」を付け忘れている場合
2. プログラムの記述ミスの場合

   があります。

1. 実際に「；」を付け忘れている場合

```
「7行目」で記述エラーを発見しました。
「;」を付け忘れています。
```

行の最後の「；」をつけ忘れている場合は指定された「〇行目」の一つ前のプログラムの最後につけ忘れています。

```
5    printf("Hello, world¥n")
6
```

上の例の場合は「７行目」と指摘され、探すと一つ前のプログラムの最後、「５行目」の最後に「；」をつけ忘れています。

2. プログラムの記述ミスの場合

```
「3行目」で記述エラーを発見しました。
「;」を付け忘れています。
```

この場合は｛｝の外側で記述ミスの場合です。

```
3 it main(void)
4 {
```

指摘された「３行目」の「int」が「it」になっていました。

ほかにも

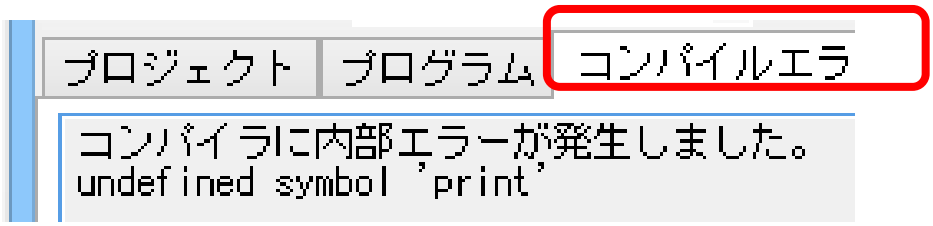
```
「5行目」で記述エラーを発見しました。
unrecognized character ¥x81
```

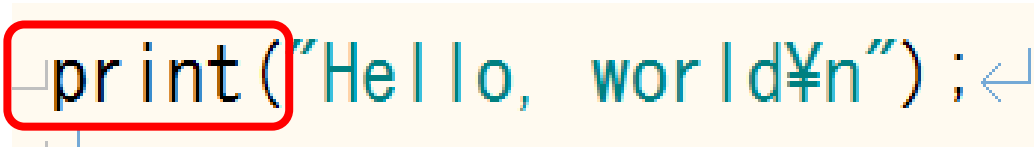
Unrecognized character　の場合はプログラム中に全角文字などのプログラムとしておかしな文字がある場合で、指摘された「〇行目」に間違いがあります。

```
5    ｐｒｉｎｔｆ("Hello, world¥n");
```

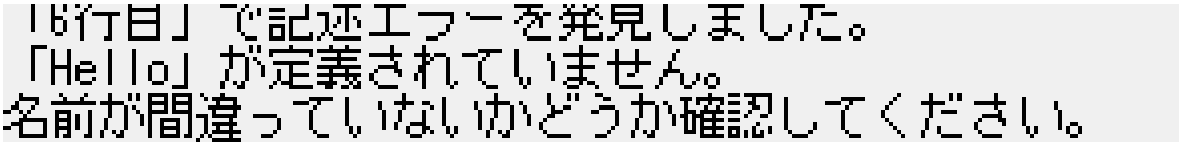
上の例の場合は「５行目」の「ｐｒｉｎｔｆ」が全角です。

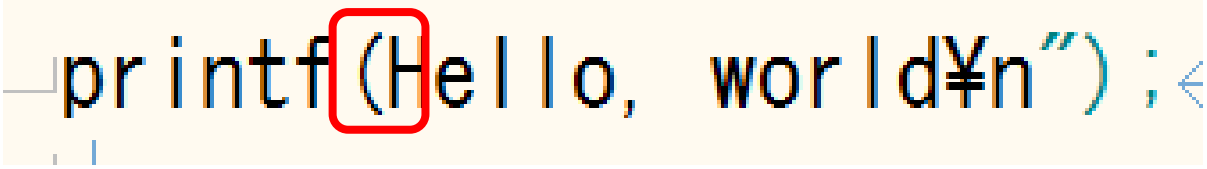
・記述が間違っている場合　例「printf」を「print」



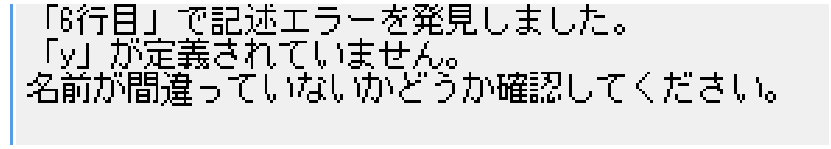


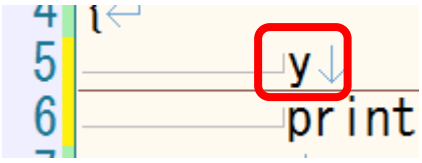
・　「　″　」の書き忘れなどの記述ミス





・定義されていないプログラムが記述されている

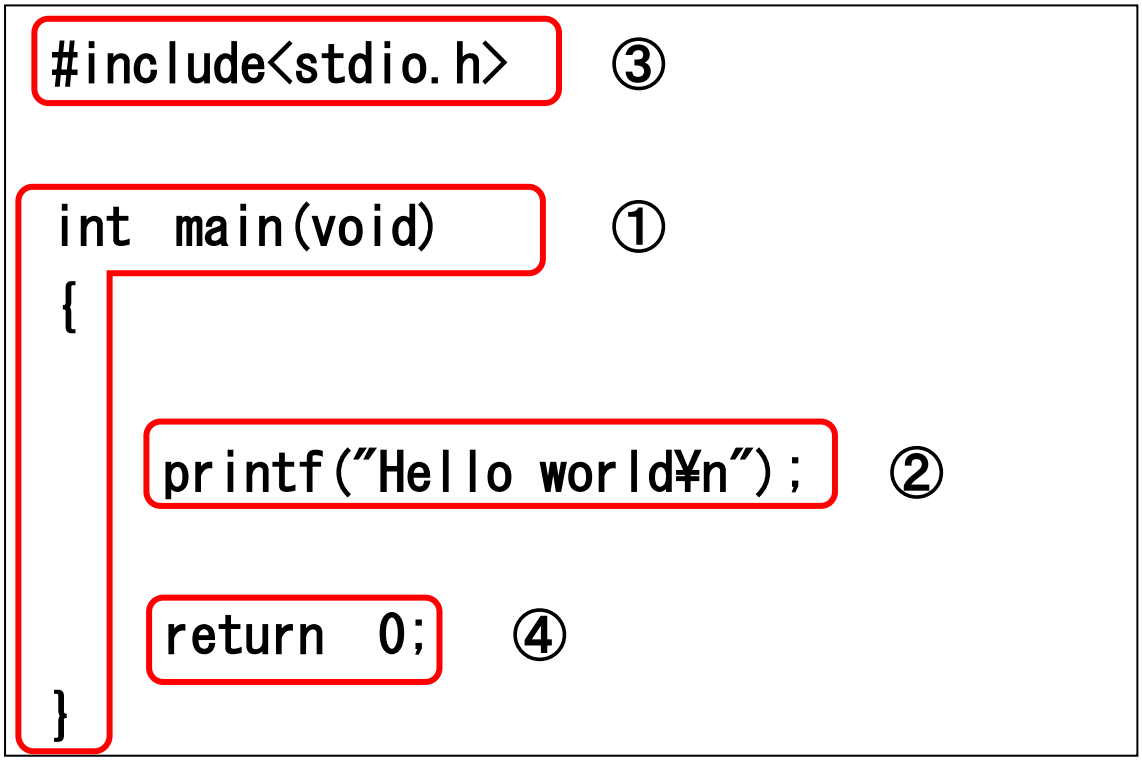




など、エラー個所を探して修正→実行→修正→・・・と繰り返しましょう。

5. 1 つ 1 つの文章の意味

前章で作成したプログラムは最も初歩的なプログラムといえます。そのため、どのようなプログラムを作る際にも、必ず記載しなくてはならない文章があります。この章では、それぞれの文章の意味について簡単に解説します。



① int main(void){ }

「コンピュータにさせたい処理内容はここからここまでですよ」ということを明示的にするための記述です。{ }の中に書いてある文章の一番上から順番に処理をしていきます。main 関数と呼ばれ、どのようなプログラムでも必ず書きます。なぜ main 関数が呼ばれるのでしょうか？そもそも関数とはどういう意味でしょうか？順を追って説明します。

まず、プログラムにおける関数というのは、「コンピュータにさせる処理内容を分割した単位」のことを指します。プログラムによっては、何万行という膨大な文章が記述されます。 そのため、いくつかの固まりに分割して書いた方が見やすいですよね。見やすいようにした方が、プログラミングの際に記述ミスをする可能性が低くなりますし、第三者がプログラムを解読する場合、読みやすいに決まっています。そのため、膨大な行数のプログラムを作る際には、関数と呼ばれる単位に分割し、それぞれを結合することでコンピュータにさせたい処理内容を完成させます。

小説でいうところの「章」に相当します。小説では必ず読者に伝えたい「テーマ」がありますよね。そして、そのテーマを読者に伝えるために、「章」という単位に分割しています。これとほぼ同じです。小説における「テーマ」は、プログラムにおける「処理内容」に相当し、「章」は「関数」に相当します。

章には順番があるように、関数にも実行する順番があります。その順番をコンピュータはどうやって判断しているのでしょうか？ここで main 関数が関係してきます。プログラムは必ず main 関数から始まり、main 関数から様々関数を順番に呼び出します。プログラムにおける中心的な役割を持つ関数なので、main 関数と呼ばれます。

②　printf(″Hello　world¥n″);

「Hello world と画面に表示しろ」という記述です。文字列の最後「¥n」は改行を意味します。「printf(“文字列¥n”);」と書くだけで、文字列を画面に表示させることができます。頻繁に利用される文章の１つです。

″″これらの中では全角でもプログラムとして成り立ちます

しかし、皆さん疑問に思いませんか？「文字列を画面に表示させる」という処理内容が printf というたった 1 行だけでいいなんて都合が良すぎます。これにはある秘密があります。皆さんが最初に書いた謎の記述③がその秘密を握っているのです。

③　#include<stdio.h>

よく「おまじない」と言われる記述です。かなり簡略しますが、できるだけわかりやすく説明します。 この一行は、「頻繁に使うと考えられる処理（printf 以外も含む）」を 1 行で使えるようにするための記述です。よく使う処理を 1 行で使えるようにプログラムに含みますという意味で、「include」という単語が使われています。

実を言うと、printf のように「文字列を画面に表示させる」という処理内容を自分で書こうとする場合、熟練のプログラマーでも頭を悩ませるレベルです。それを 1 行で使えるというのはとても便利ですよね。

④　return 0;

プログラムの終了を明示的にするための記述ですが、この 1 行はあってもなくてもいいです。なぜならば、①によって（「 } 」がプログラムの最後を意味している）、どこまでがコンピュータにさせる処理内容なのかはわかっているからです。ただ、④を書かない場合、綺麗な終わり方ではないと言われます。

皆さん、コンピュータをシャットダウンする時どうしていますか？電源ボタンを長押しする「強制終了」をしていませんよね。強制終了してもいいですが、保存していたデータが消えるなどのトラブルのもとになるような行為はしない方がいいです。皆さん「スタートボタン」から「シャットダウン」という正規のやり方を採用しているはずです。それと全く同じです。④を書かないでプログラムの終了をしてもいいですが、何らかのトラブルのもとになる可能性があります。そのようなことがないように、④をしっかりと書いて、正規の終了をするようにします。

6. 変数を使ってみよう

以下のプログラムを書きましょう。

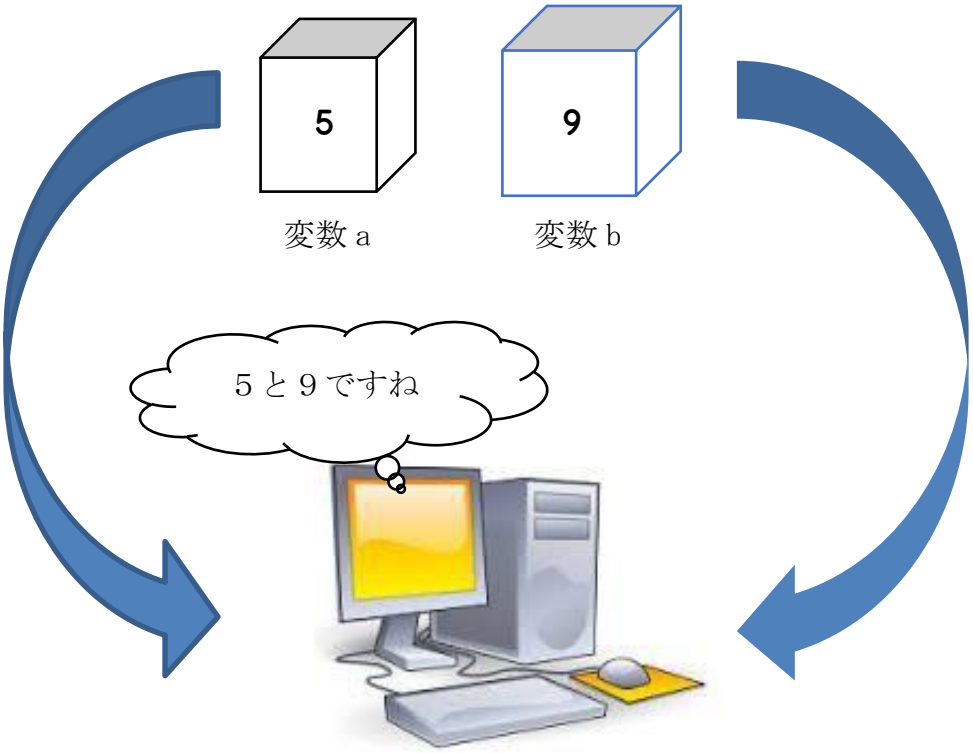
```
#include<stdio.h>
int main(void){

    int a = 5;
    int b = 9;
    int c;

    c = a + b;

    printf("整数%d + 整数%d は%d です¥n",a,b,c);

    c = a - b;

    printf("整数%d - 整数%d は%d です¥n",a,b,c);

    return 0;
}
[EOF]
```

できた人は a,b の値を変えてみましょう。

変数について

プログラムを書く上で、数値を扱いたい場合は多々あります。数値を直接書いてもいいですが、その数値を変えることになったらどうなりますか？その数値が書いてある箇所を全部書き直さなければなりません。

そのような場合、数値を入れるための「箱」を用意してあげると便利です。その箱を変数と言います。コンピュータは、変数から数値を取り出して、数値を扱います。



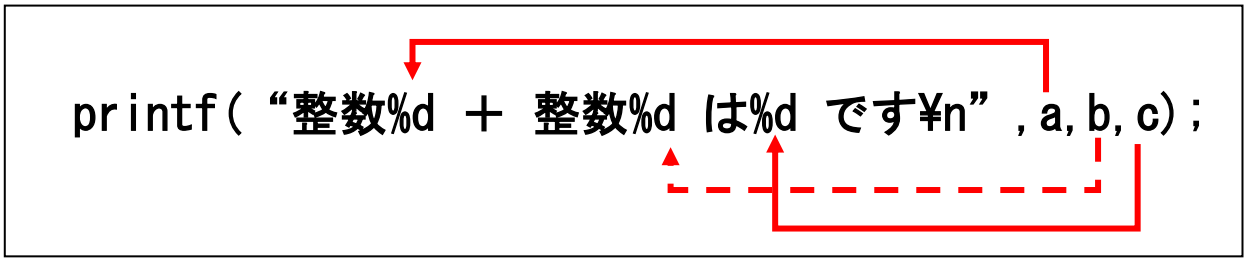
①　「変数 a、b、c を使います」という記述です。「int」は integer（整数）を意味します。数値といっても、小数など幾つかの種類がありますよね。コンピュータは自律的に考えることができないので、どの種類の数値なのかを明示的に教えてあげる必要があります。この例では整数を扱うので、整数であることを意味する「int」を添えてあげます。そして、変数 a に 5 を入れ、変数 b に 9 を入れます。変数に数値を入れることを、「代入する」と表現します。

②　変数 a と変数 b を足した結果を変数 c に代入しています。今、変数 a には 5 が入っており、変数 b には 9 が入っています。そのため、変数 c には 14 が代入されます。次に、変数 a から変数 b を引いた結果を変数 c に代入しています。当然、変数 c には -4 が代入されます。

③　先ほどの printf ですが、

```
printf(“整数%d ＋ 整数%d は%d です¥n”,a,b,c);
```

なにやらよくわからない文字「**%d**」がありますね。「%d」を文字列の中に含めることで、変数に格納されている整数を表示することができます。
%d と変数との対応関係は以下のようになっています。



つまり、左から順次対応していることになります。
実際に　a = 5, b = 9, c = a+b = 14　なので

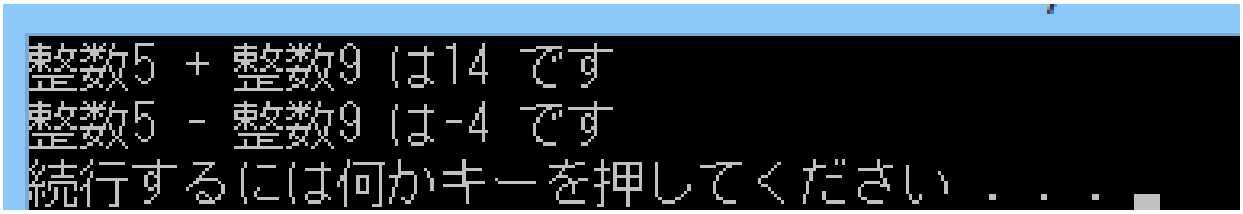
「**整数 5 ＋ 整数 9 は 14 です**」と表示されます。

一度 printf で c の値を表示した後

c の値を a+b から a-b にして再度 c の値を表示しています。

a = 5, b = 9, c = a-b = -4　なので

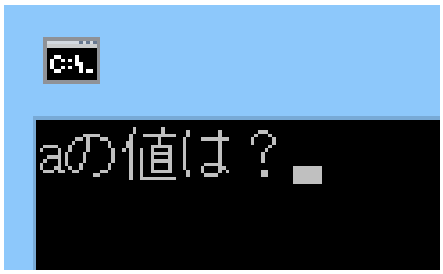
「**整数 5 － 整数 9 は -4 です**」と表示されます。



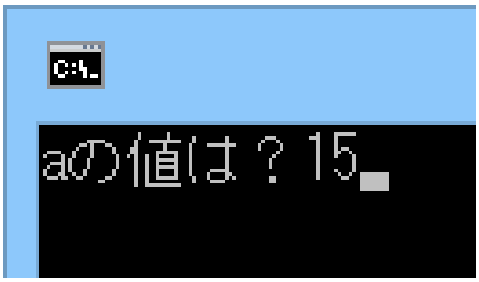
7. 入力した数字を反映させてみよう

6 で作成したプログラムに以下を書き加えてみよう。

```
7   int b = 9;
8   int c;
9
0   printf("aの値は？");
1   scanf("%d", &a);
2
3   c = a - b;
```

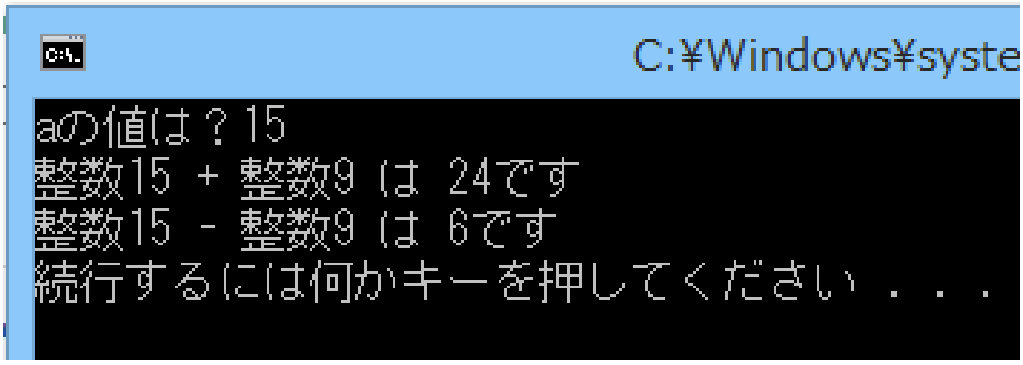


printf で指示した「a の値は？」の後にカーソルが点滅しています。



適当な数字を入力し、エンターキーを押してみましょう。

scanf



すると「入力した数字(15)+b（9）=24, および -b（9）=6

が計算されて表示されます。

```
scanf("%d", &a);
```

scanf はキーボードから入力した文字などを反映させます。

%d は printf 同様、変数に格納されている整数です。

&a は入力した値を入れる変数を指定しています。**&変数名**で指定します。
今回は a に入力した値を入れているため &a となっています

※最初に a に数値を代入していますが、そのあとの scanf で再度代入しているため
scanf の数値が反映されます。

printf と同様に複数指定することもできます。

```
scanf("%d%d", &a, &b)
```

この例の場合、1 つ目の%d が a の変数に、2 つ目の%d が b の変数になります。

発展問題

b の値も入力する形にしましょう。
a と同じように値を問う形式にしましょう。

8. 条件文を使ってみよう

計算結果によって表示を変えるプログラムを作りましょう。

以下のプログラムを書きます。日本語以外はすべて半角英数字です。
日本語部分や行数までまったく同じである必要はありません。

```
#include <stdio.h>
↓
int main(void)↓
{↓
    int a,b,c;↵
    ↵
    printf("aの値は？");↵
    scanf("%d",&a);↵
    ↵
    printf("bの値は？");↵
    scanf("%d",&b);↵
    ↵
    c = a - b ;↵
    ↵
    if (c>0){↵
        printf("計算結果は正の数 %d です。¥n",c);↵
    }↵
    else if(c<0){↵
        printf("計算結果は負の数 %d です。¥n",c);↵
    }↵
    else{↵
        printf("計算結果は %d です。¥n",c);↵
    }↵
    ↓
    return 0;↵
    ↓
} [EOF]
```

if

① 変数 c に代入されている値が正の場合、「計算結果は正の数~です」と表示し、変数 c に代入されている値が負の場合、「計算結果は負の数~です」と表示させています。このように、ある条件によって動作を変えたい時に使うのが「if 文」です。if 文は以下の形式に基づいて記述します。

```
if(条件式①){
条件式①が真（成り立っている)」の場合の処理
}
else if（条件式②){
条件式①が偽（成り立っていない）が
条件式②が真（成り立っている)」の場合の処理

」の場合の処理
}
 else if（条件式③){
条件式①、②が偽（成り立っていない）が
条件式③が真（成り立っている)」の場合の処理
}
 else if（条件式④）{
・・・・※else if は繰り返すことできます
}
 else{
条件式①,②,③,④,・・・がすべて「偽（成り立っていない)」の場合の処理
}
```

例

c = a-b = 5-9 =-3

「条件式①　c>0 である」ではない

「条件式②　c<0 である」なので

条件式②の場合の処理を実行します。

> 大なり, < 小なり、以上、以下、イコール、イコールではない、は以下のように記述します。イコールの場合＝を 2 つ続けるのを忘れないようにしましょう

a > 1	⇒ a は 1 より大きい
a >= 1	⇒ a は 1 以上
a < 1	⇒ a は 1 より小さい
a <= 1	⇒ a は 1 以下
a == 1	⇒ a は 1 と等しい
a != 1	⇒ a は 1 と等しくない

9. 条件の組み合わせ

「＆＆」と「｜｜」

「｜」の入力方法は shift キー ＋ ¥　　¥ はキーボード Backspace の横

```
if (a>0&& b>0){
    printf("aもbも正の値です。¥n",c);
}
else if(a>0 || b>0){
    printf("どちらかは正の値です。¥n",c);
}
else if(a<0 || b<0){
    printf("どちらかは負の値です。¥n",c);
}
else{
    printf("aもbも0です。¥n",c);
}
```

「 X && Y」
Xの条件とYの条件を両方満たす場合
(a>0 && b>0)

の場合 a>0 であり b>0 ということ

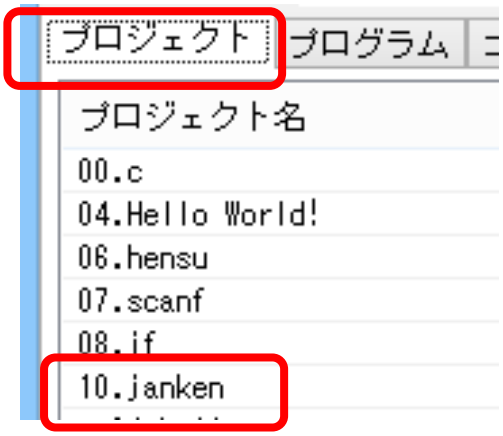
たとえば a=5, b=9 なら ok ですが
a=5, b=-9 や a=0, b=9 などどちらかが正の値ではない場合は条件から外れます。

「 X || Y」
Xの条件か、Yの条件を満たす場合
(a>0 || b>0)

の場合　a>0 または b>0 ということになります。
a=5, b=9 でも、a=5, b=-9 でも a=-5, b=9 でも a=5, b=0 でも OK です。
どちらかがマイナスの場合 a=-5, b=0 や a=-5, b=-9 は条件から外れます。

10. 演習問題

以下のプログラムを開き「**じゃんけんプログラム**」を作成して下さい。
今まで使っていたタブ「プログラム」の横に「プロジェクト」というタブがあります。
その中に「10. janken」というプロジェクトがありますのでそちらをダブルクリックして開きます。開いたらプログラムに戻ってください。



問題
あなたとコンピュータでじゃんけんをします。

すでに書いてあるプログラム
- コンピュータの出す手を 「 c 」という因数であらわしています。
- コンピュータの手はランダムで出ます。
- 1 がグー、2 がチョキ、3 がパーとします。

- ✓ あなたが入力した手を「 a 」としてください。 scanf～
- ✓ 勝ち負けを考え、if および else if を繰り返し使ってあなたの勝ち負けを判定してください。※else if を何回も繰り返します。
- ✓ ひねくれもののあなたが 1,2,3 以外を入力する場合も考えましょう。

発展問題

あなたが何を出して、コンピュータがなにをだして、勝った、まけた、あいこの文章まで書きましょう

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

//コンピュータの出す手を決めている
int comp() {
 srand(time(NULL));
 int c = rand()%3+1;

 return c;
}
int main(void){
int a, c;
c = comp();

 //ここから入力必要個所

        printf("手を入力してください [1:グー 2:チョキ 3:パー] ");
        scanf("",&);

        if()
        {
                printf("");
        }
        else if ( )
        {
        }
        else{

        }

 return 0;
}
```

すでに書いてあるプログラム
・ コンピュータの出す手を 「 c 」という因数と定義して
c は 3 つの数字 0, 1, 2 に＋1 した数字のランダムとする。

11. 最後に

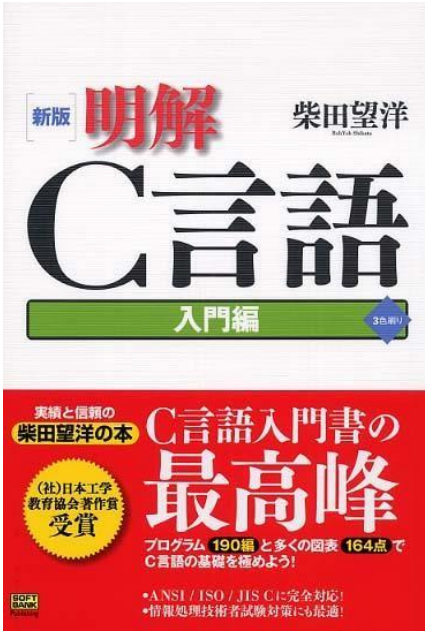
ここで解説した C 言語の文章は一部ですが、この講習で記述した文章がもつ意味を理解した皆さんならば、他の文章も簡単に理解できるはずです。プログラミングは何度もプログラムを書かなければ忘れてしまいます。この講習をきっかけに、引き続きプログラミングを学習して下さることを期待します。

付録

独学で C 言語を学びたい方は、右の本がおすすめです。

もちろんほかにもたくさん本があります。

Web 上にもたくさんの紹介ページがありますので



この講習で利用した「Easy IDEC」は家のパソコンに無料でインストールすることが可能です。以下のサイトを参考にインストールを行ってください。

http://9cguide.appspot.com/p_9cide.html

演習課題の解答

「Easy IDEC」のプロジェクトに各例文があります。

注意

和泉キャンパスメディア棟の PC の場合デスクトップに保存したものはログオフ後に消去されます。
MyDocs (X)ドライブは消去されないので MyDocs に保存する習慣をつけましょう。