理科研究（物理分野）

（2015 年度春学期）

【参考】

小学1年生にプログラミング教育--武雄市、DeNA、東洋大

2015年2月13日

佐賀県武雄市、ディー・エヌ・エー（DeNA）、東洋大学は 2 月 12 日、武雄市立山内西**小学校の 1 年生** 40 人を対象に、全 8 回実施したプログラミングの授業についての成果報告会を開催。授業で児童たちが作成したコンテンツを披露した。（中略）

授業では、タブレット端末にプログラミングアプリをインストールして、「キャラクターを動かす」コンテンツの作り方を指導した。プログラミングアプリは、川崎氏がプログラミング言語「Scratch」をベースに、タブレット端末でのタッチ操作に特化した仕様で開発した。（中略）

同日、第8回の授業として、児童たちがプログラミングアプリで作ったコンテンツを1人ひとり発表。「サンタクロースが家に入ると灯りがつく」「UFOに爆弾が当たると消える」「猫がミルクを飲むと大きくなる」など、それぞれが考えたストーリーを表現した。

南場氏は、「**小学1年生がどこまでタブレットを使いこなせるのか心配したが、全く問題なく第1回目の授業からプログラミングの講義をすることができた**」と説明。児童たちの発表会を見た感想として、「1つとして同じ作品がなかった。1人1台の端末で、それぞれが個を表現できたのではないか」と語った。（中略）

教えられますか？

一方で、「**小学3～4年生くらいになれば、コードを書くようなプログラミング教育も可能になる**。その場合は、キーボード付きのデバイスの方が良いかもしれない」とも考える。川崎氏によれば、今回開発したアプリは、OSによらずタブレット端末で動作する仕様になっており、iPadやWindowsタブレットへの展開も可能だとする。

指導内容については、小学1年生の発達を考慮した改善が必要のようだ。今回の授業では、「～を～かいくりかえす」「もし～なら」など、ループや条件分岐まで指導内容に含めた。学級担任の藤瀬澄子氏は「ブロックをつなげる作業は1年生には難しく、児童1人ではできなかった。**1対1のサポートが必要だった**」と語った。（以下略）

サポートできますか？

ZDNet Japan のウェブサイトより

アプリケーションのインストール

※ここでは、WindowsＰＣ上にプログラミング環境を構築する方法を解説する。おもにWindows 7を想定した説明になっているが、Windows 8やWindows XPでも同様にインストールできる。

【アカウントの確認】

ＰＣ上での自分のユーザ名（ログインするときの名前）を確認する。ユーザ名が半角英数字でない場合はインストールできない。ユーザアカウントを作成したときには半角英数字でなかったが、後で半角英数字に変更した場合も同様である。ユーザ名が半角英数字でない場合は、半角英数字のユーザ名で新しいアカウントを作成し、そのアカウントでインストールやプログラミングを行なう必要がある。

例：○ hgakugei　　× 学芸花子　　× ｈｇａｋｕｇｅｉ　（←全角英数字）

× 学芸花子からhgakugeiに変更

また、インストール作業では管理者権限が必要である。

【インストール作業】

ダウンロードした圧縮フォルダ「物理分野のアプリケーション」に保存されている圧縮フォルダmingw.zipをＰＣにコピーした後、すべて展開して、中身を確認する。

☞mingw.zipを右クリックして「すべて展開」を選択する。

単にダブルクリックしただけでは一時展開なので以下の作業がうまくいかない。

①フォルダMinGWをC:¥にコピーする。

☞スタートボタンからコンピュータを選び、Ｃドライブをダブルクリックすると、C:¥を開くことができる。

②フォルダemacs-24.3をC:¥にコピーする。

③フォルダgnuplotをC:¥にコピーする。

④システム環境変数を設定する：

スタートボタン→コントロールパネル→システムとセキュリティ→システム

→システムの詳細設定→詳細設定→環境変数

「システム環境変数」の中の「Path」を選択する。

「編集」をクリックする。

「；」で区切りながら、「C:¥MinGW¥bin;C:¥emacs-24.3¥bin;C:¥gnuplot¥bin」を追加する。

☞すでに書かれている内容の末尾にセミコロンを付け加え、その後にかぎ括弧の内容を追記する。

注意：すでに設定されている内容を絶対に変えてはいけない。大文字と小文字の違い、コロンとセミコロンの違いにも注意すること。この部分は特に慎重に行なう必要がある。

以上の設定変更を反映するため、①～③の作業で開いていたウィンドウを閉じて、新たに別のウィンドウを開いて⑤以下の作業を進める。

⑤C:¥MinGW¥msys¥1.0¥msys.bat をダブルクリックして実行する。
ターミナルが出てきたら以下のコマンドを実行する。

```
$ exit ↵          ☞exit とタイプして、エンターキーを押す
```

⑥ショートカットを作成する：
C:¥MinGW¥msys¥1.0¥msys.bat へのショートカットと
C:¥MinGW¥msys¥1.0¥home¥*username* へのショートカットを
デスクトップ上に作成する。
☞*username* は(半角英数字の)ユーザ名のことです。

⑦フォルダ files の中のファイル profile と emacs を
C:¥MinGW¥msys¥1.0¥home¥*username*¥の中にコピーする。

⑧ショートカットから msys を開始する。
☞「 msys.bat へのショートカット 」をダブルクリックする。
ターミナルが出てきたら以下のコマンドを実行する。

```
$ mv   profile   .profile ↵          ☞３つめの単語の語頭はピリオドである
$ mv   emacs   .emacs ↵
$ exit ↵
```

以上でインストールは完了である。再び「 msys.bat へのショートカット 」をダブルクリックすればターミナルが起動し、Emacs や gnuplot を利用できるはずである。

アンインストールについて
①～③で作成したフォルダを削除し、④で行なった環境変数の変更を元に戻せば、インストール前の状態に戻すことができる。

第１部　放物運動のシミュレーション

プログラミング作業の流れ

まずターミナルを起動する。　（⇒簡単操作マニュアル１）

☞ターミナル（端末）がすべての作業の起点になります。

１．Emacs でプログラムを書く。　（⇒１－②、２－②）

２．コンパイルする。　（⇒１－③）

３．実行する。　（⇒１－④）

４．結果を確かめる。　（⇒１－⑤⑥）

簡単操作マニュアル

１．ターミナルの使い方

ターミナルを起動する	デスクトップの「 msys.bat へのショートカット 」を ダブルクリック
①ファイルの一覧を表示する	$ ls ↵
ファイルをコピーする	$ cp *コピー元ファイル名* *コピー先ファイル名* ↵
ファイル名を変更する	$ mv *変更前のファイル名* *変更後のファイル名* ↵
ファイルを消去する	$ rm *ファイル名* ↵
②Ｅｍａｃｓを起動する	$ emacs *ファイル名* & ↵ （または emacs & ↵）
③コンパイルする	$ cc *ファイル名* ↵ math.h を使う場合： $ cc *ファイル名* -lm ↵
④実行する	$ ./a.exe ↵
⑤ファイルの中身を見る	$ less *ファイル名* ↵ 矢印キーで移動、Ｑを押して終了
⑥ｇｎｕｐｌｏｔを起動する	$ gnuplot ↵
ターミナルを終了する	$ exit ↵

２．Ｅｍａｃｓの使い方

①ファイルを開く	コントロールキー＋Ｘ コントロールキー＋Ｆ →ファイル名を入力
②ファイルを保存する	コントロールキー＋Ｘ コントロールキー＋Ｓ
※操作の取り消し	コントロールキー＋Ｘ コントロールキー＋Ｇ
Ｅｍａｃｓを終了する	コントロールキー＋Ｘ コントロールキー＋Ｃ

３．ｇｎｕｐｌｏｔの使い方

①グラフをかく	> plot ”*ファイル名*” ↵ 複数の場合： plot ”*ファイル名*”， ”*ファイル名*”， … ↵
②折れ線グラフにする	> set style data lines ↵
範囲を指定する	> set xrange [*0*:*50*] ↵ > set yrange [*0*:*30*] ↵
③再描画する	> replot ↵
※画像ファイルに保存する	画面上で作図したあとで： > set term png ↵ > set output ”*ファイル名.png*” ↵ > replot ↵
ｇｎｕｐｌｏｔを終了する	> quit ↵

プログラムの基本構造

```
#include  <stdio.h>
#include  <math.h>      /*  この行は必要な場合※のみ  */
int  main  (void)
{

/*  ここに処理の内容を書く  */

    return  0;
}
```

※平方根や三角関数などを使う場合

①滞空時間と飛距離を計算しよう

作業の手順：以下のサンプルプログラムを Emacs で作成しよう。

☞プログラムのファイル名は自由だが、「.c」で終わる必要がある。
　例：prog01.c なら　　⇒　　ターミナル上で　$ emacs　prog01.c　&↵

☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。
コントロールキーを押しながらＸを押す

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

    float g, V, angle, theta, T, X;          /* 変数を宣言する */

    g = 9.8;                                 /* 定数を代入する */

    printf ("Velocity [m/s], angle [deg.]?\n");
    scanf ("%f,%f", &V, &angle);             /* 条件を入力する */
    theta = 3.14159 / 180.0 * angle;

    T = 2.0 * V * sin (theta) / g;
    X = V * cos (theta) * T;

    printf ("T = %9.3f, X = %9.3f\n", T, X);    /* 結果を出力する */

    return 0;
}
```

この部分が計算式です。

※　大文字と小文字は区別する。
※　個々の処理はセミコロン「 ; 」で区切る。
※　/* …… */となっている部分はコメントであり処理に影響しないので、書かなくてよい。
※　環境によってはバックスラッシュ「 \ 」が円記号「 ¥ 」で表示される場合があるが問題ない。

メモ：

$$T = \frac{2w_0}{g} = \frac{2V\sin\theta}{g}$$

$$X = u_0 T = \frac{2u_0 w_0}{g} = \frac{2V^2 \sin\theta\cos\theta}{g} = \frac{V^2 \sin 2\theta}{g}$$

実行例：ターミナル上でコンパイルして実行してみよう。

```
$  cc  prog01.c  -lm ↵          コンパイルします。

$  ls ↵                        実行ファイルを確認します。

a.exe  prog01.c

$ ./a.exe ↵                    実行します。

Velocity  [m/s],  angle  [deg.]?

10,45 ↵                        数値を入力します。

T  =     1.443,  X  =    10.204   結果が出力されます。
```

メモ：

$V = 10\,[\mathrm{m}/\mathrm{s}]$、$\theta = 45^\circ$　⇒　$T \fallingdotseq 1.4\,[\mathrm{s}]$、$X \fallingdotseq 10\,[\mathrm{m}]$

☞正常に実行できることを確認したら、Emacs を終了してよい
（コントロールキー＋Ｘ　コントロールキー＋Ｃ）。

②初期の状態を計算しよう

作業の手順：①で作成したプログラムをコピーして、Emacs で完成させよう。

☞コピーするときには cp コマンドを使う。
　例：$ cp　prog01.c　prog02.c ↵　　（⇒簡単操作マニュアル１－①）
☞コピーしたファイルを改めて emacs で開く。

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

    float  g,  V,  angle,  theta,  x,  z,  u,  w;  /*  変数を宣言する  */

    g  =  9.8;                                     /*  定数を代入する  */

    printf  ("Velocity  [m/s],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &V,  &angle);                /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;

    x  =  0.0;                                     /*  初期値を計算する  */
    z  =  0.0;
    u  =  V  *  cos  (theta);                          ここで初期値を計算します。
    w  =  V  *  sin  (theta);

    printf  ("%9.3f  %9.3f\n",  x,  z);            /*  結果を出力する  */

    return  0;
}
```

実行例：ターミナル上でコンパイルして実行してみよう。

$ cc prog02.c -lm ↵	コンパイルします。
$ ./a.exe ↵	実行します。
Velocity [m/s], angle [deg.]?	
10,45 ↵	数値を入力します。
0.000 0.000	結果が出力されます。

③0.01 秒後の状態を計算しよう

作業の手順：②で作成したプログラムをコピーして、Emacs で完成させよう。

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

    float  g,  V,  angle,  theta,  x,  z,  u,  w,
           dxdt,  dzdt,  dudt,  dwdt;              /*  変数を宣言する  */

    g  =  9.8;                                     /*  定数を代入する  */

    printf  ("Velocity  [m/s],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &V,  &angle);                /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;

    x  =  0.0;                                     /*  初期値を計算する  */
    z  =  0.0;
    u  =  V  *  cos  (theta);
    w  =  V  *  sin  (theta);

    printf  ("%9.3f  %9.3f\n",  x,  z);            /*  結果を出力する  */

    dxdt  =  u;                                    /*  時間微分を計算する  */
    dzdt  =  w;
    dudt  =  0.0;
    dwdt  =  -  g;

    x  =  x  +  0.01  *  dxdt;                     /*  次の時刻の値を計算する  */
    z  =  z  +  0.01  *  dzdt;
    u  =  u  +  0.01  *  dudt;
    w  =  w  +  0.01  *  dwdt;

    printf  ("%9.3f  %9.3f\n",  x,  z);            /*  結果を出力する  */

    return  0;
}
```

ここで 0.01 秒後の値を計算します。

実行例：ターミナル上でコンパイルして実行してみよう。

```
$  cc  prog03.c  -lm ↵          コンパイルします。

$ ./a.exe ↵                    実行します。

Velocity  [m/s],  angle  [deg.]?

10,45 ↵                        数値を入力します。

     0.000       0.000         結果が出力されます。
     0.071       0.071
```

④結果をファイルに書き出そう

作業の手順：③で作成したプログラムをコピーして、Emacs で完成させよう。

実行したら出力ファイルの内容を確認しよう。

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

    float  g,  V,  angle,  theta,  x,  z,  u,  w,
           dxdt,  dzdt,  dudt,  dwdt;              /*  変数を宣言する  */
    FILE  *fp;


    g  =  9.8;                                     /*  定数を代入する  */


    printf  ("Velocity  [m/s],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &V,  &angle);                /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;


    x  =  0.0;                                     /*  初期値を計算する  */
    z  =  0.0;
    u  =  V  *  cos  (theta);
    w  =  V  *  sin  (theta);


    fp  =  fopen  ("output.txt",  "w");           /*  ファイルを開く  */
    fprintf  (fp,  "%9.3f  %9.3f\n",  x,  z);      /*  結果をファイルに出力する  */


    dxdt  =  u;                                    /*  時間微分を計算する  */
    dzdt  =  w;
    dudt  =  0.0;
    dwdt  =  -  g;


    x  =  x  +  0.01  *  dxdt;                     /*  次の時刻の値を計算する  */
    z  =  z  +  0.01  *  dzdt;
    u  =  u  +  0.01  *  dudt;
    w  =  w  +  0.01  *  dwdt;


    fprintf  (fp,  "%9.3f  %9.3f\n",  x,  z);      /*  結果をファイルに出力する  */


    fclose  (fp);                                  /*  ファイルを閉じる  */
    return  0;
}
```

実行例：ターミナル上でコンパイルして実行してみよう。

```
$  cc  prog04.c  -lm ↵          コンパイルします。

$ ./a.exe ↵                    実行します。

Velocity  [m/s],  angle  [deg.]?

10,45 ↵                        数値を入力します。

$  ls ↵                       出力ファイルを確認します。

a.exe  output.txt  prog01.c  prog02.c  prog03.c  prog04.c

$  less  output.txt ↵          出力ファイルの中身を見ます。→見終わったらＱで終了。
```

⑤10 秒後まで計算しよう

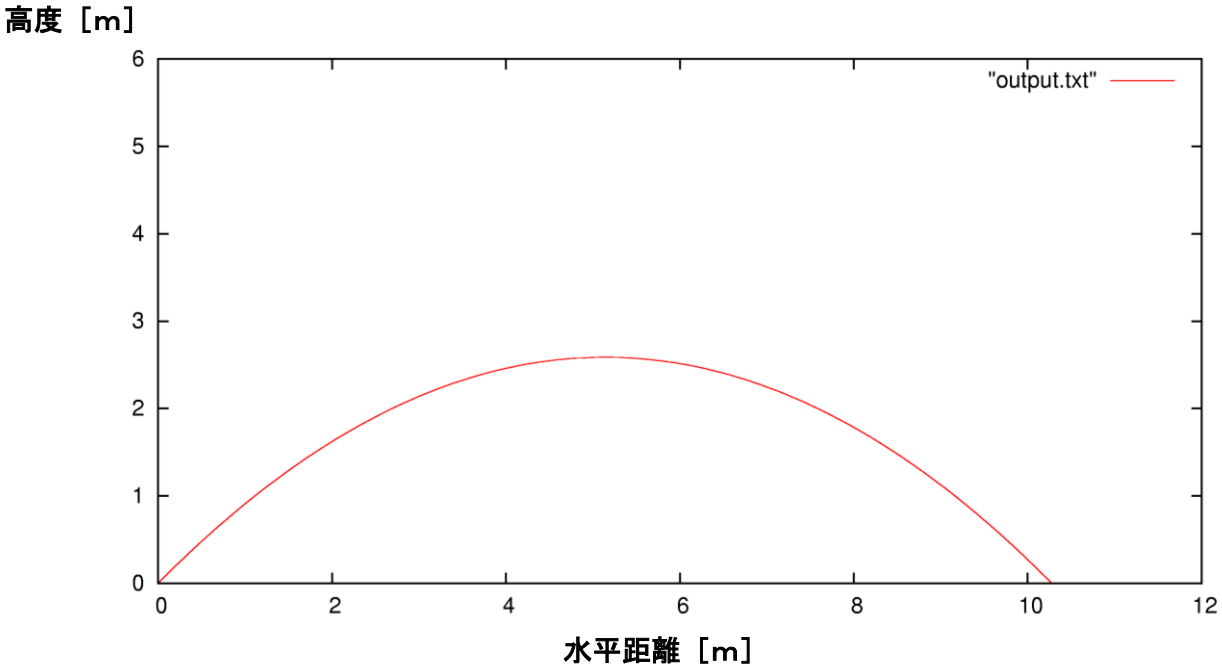
作業の手順：④で作成したプログラムをコピーして、Emacs で完成させよう。

実行したら出力ファイルの内容を gnuplot で作図しよう。

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


    int  i;
    float  g,  V,  angle,  theta,  x,  z,  u,  w,
           dxdt,  dzdt,  dudt,  dwdt;              /*  変数を宣言する  */
    FILE  *fp;

    g  =  9.8;                                     /*  定数を代入する  */

    printf  ("Velocity  [m/s],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &V,  &angle);                /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;

    x  =  0.0;                                     /*  初期値を計算する  */
    z  =  0.0;
    u  =  V  *  cos  (theta);
    w  =  V  *  sin  (theta);

    fp  =  fopen  ("output.txt",  "w");            /*  ファイルを開く  */
    fprintf  (fp,  "%9.3f  %9.3f\n",  x,  z);      /*  結果をファイルに出力する  */

    for  (i=1;  i<=1000;  i++)                     /*  同じ処理を 1000 回繰り返す  */
    {

        dxdt  =  u;                                /*  時間微分を計算する  */
        dzdt  =  w;
        dudt  =  0.0;
        dwdt  =  -  g;

        x  =  x  +  0.01  *  dxdt;                 /*  次の時刻の値を計算する  */
        z  =  z  +  0.01  *  dzdt;
        u  =  u  +  0.01  *  dudt;
        w  =  w  +  0.01  *  dwdt;

        fprintf  (fp,  "%9.3f  %9.3f\n",  x,  z);  /* 結果をファイルに出力する */

    }

    fclose  (fp);                                  /*  ファイルを閉じる  */
    return  0;
}
```

作図例：実行した後、less で結果を確認したら、gnuplot で作図しよう。

$ gnuplot ↵	gnuplot を起動します。
gnuplot> set style data lines ↵	折れ線グラフを指定します。
gnuplot> plot "output.txt" ↵	グラフをかきます。
gnuplot> set xrange [0:12] ↵	範囲を指定します。
gnuplot> set yrange [0:6] ↵	
gnuplot> replot ↵	再描画します。



計算結果の例（初速度の大きさ 10m/s、仰角 45° の場合）

⑥初速度の大きさを変えてみよう

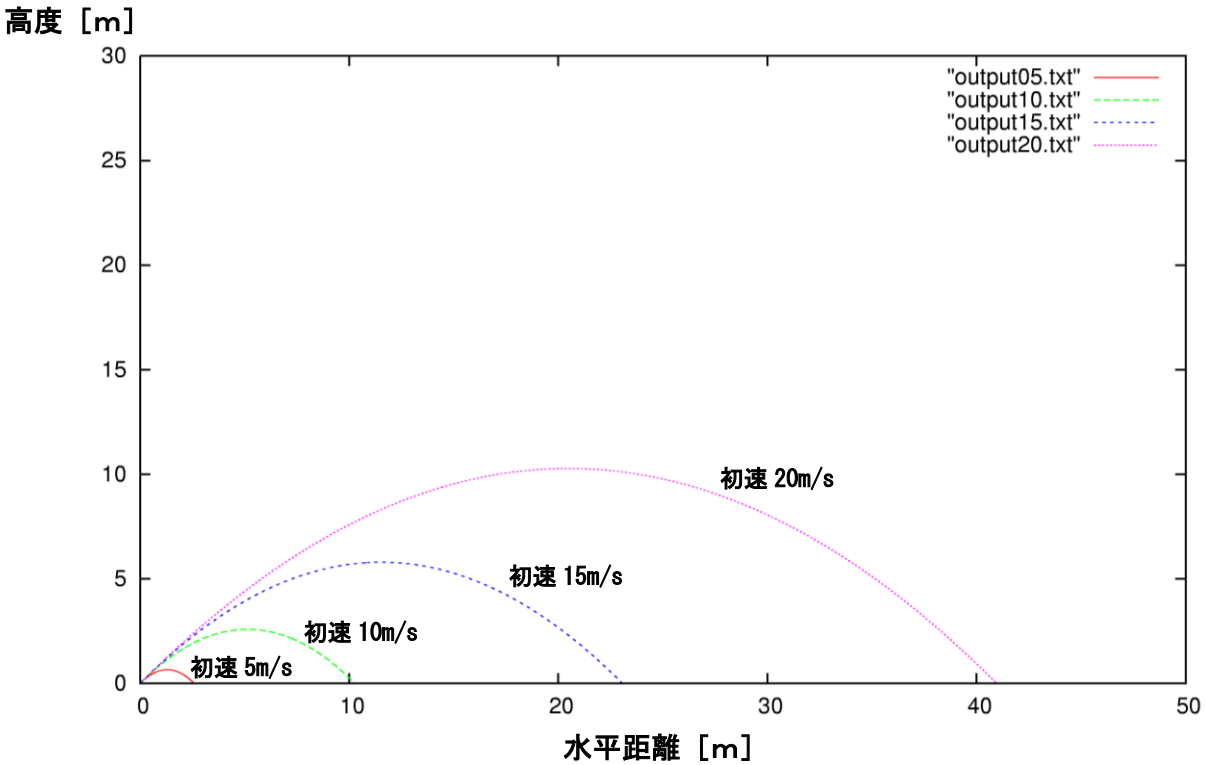
作業の手順：⑤で作成したプログラムで、仰角は一定のまま、
初速度の大きさを変えて計算しよう。
計算ごとに出力ファイル名を変えて保存しよう。
出力ファイルの内容をまとめて gnuplot で作図しよう。

初速度の大きさ 5m/s、仰角 45° で実験 → 出力ファイル名を output05.txt に変更。
初速度の大きさ 10m/s、仰角 45° で実験 → 出力ファイル名を output10.txt に変更。
初速度の大きさ 15m/s、仰角 45° で実験 → 出力ファイル名を output15.txt に変更。
初速度の大きさ 20m/s、仰角 45° で実験 → 出力ファイル名を output20.txt に変更。
☞ファイルの名前を変更するときには mv コマンドを用いる（簡単操作マニュアル参照）。

例： $ ./a.exe ↵
Velocity [m/s], angle [deg.]?
5,45 ↵
$ mv output.txt output05.txt ↵

→ gnuplot で４つの出力ファイルをまとめて作図（簡単操作マニュアル参照）。

例： $ ./a.exe ↵
gnuplot> set style data lines ↵
gnuplot> plot "output05.txt", "output10.txt", … ↵



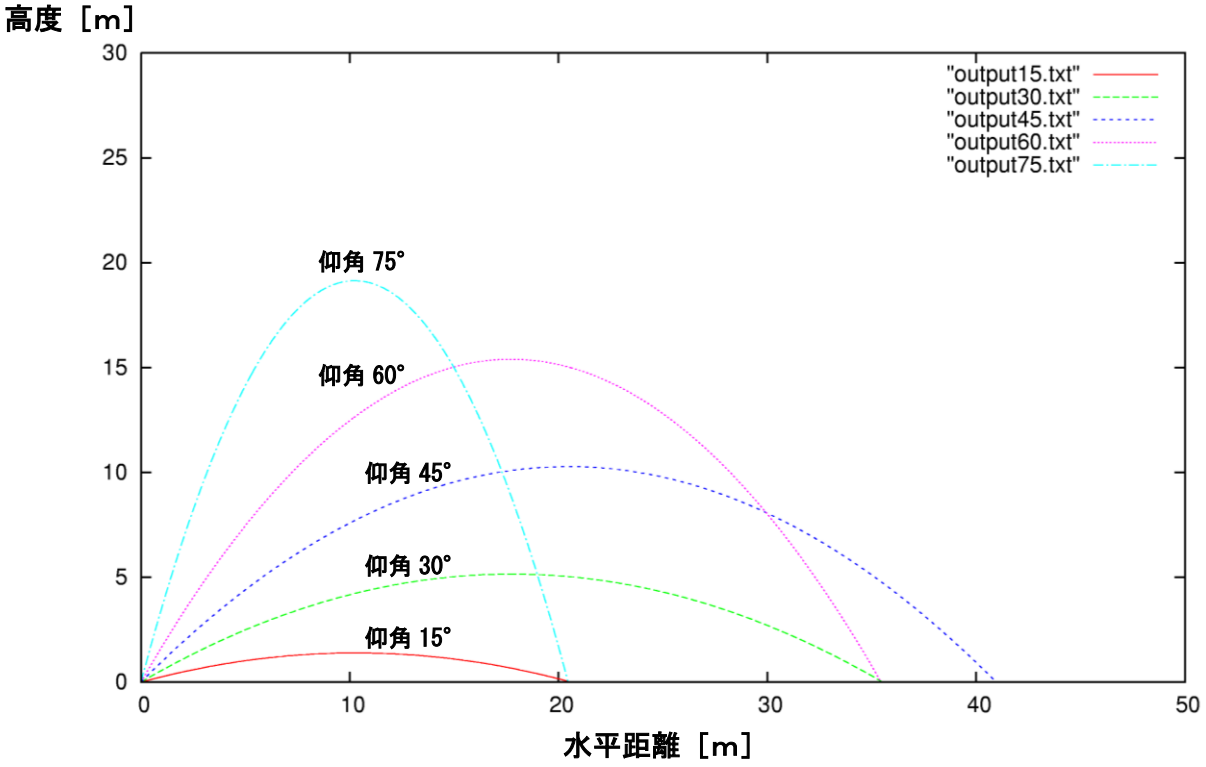
計算結果の例（仰角 45° の場合）

→ 初速度が大きいほど飛距離が長くなる。

⑦仰角を変えてみよう

作業の手順：⑤で作成したプログラムで、初速度の大きさは一定のまま、
仰角を変えて計算しよう。
計算ごとに出力ファイル名を変えて保存しよう。
出力ファイルの内容をまとめて gnuplot で作図しよう。

初速度の大きさ 20m/s、仰角 15° で実験　→　出力ファイル名を output15.txt に変更。
初速度の大きさ 20m/s、仰角 30° で実験　→　出力ファイル名を output30.txt に変更。
初速度の大きさ 20m/s、仰角 45° で実験　→　出力ファイル名を output45.txt に変更。
初速度の大きさ 20m/s、仰角 60° で実験　→　出力ファイル名を output60.txt に変更。
初速度の大きさ 20m/s、仰角 75° で実験　→　出力ファイル名を output75.txt に変更。
☞ファイルの名前を変更するときには mv コマンドを用いる（簡単操作マニュアル参照）。
→ gnuplot で５つの出力ファイルをまとめて作図。



計算結果の例（初速度の大きさ 20m/s の場合）

→ 仰角が 45° のとき飛距離が最大になる。