「　ベクトル１　」

東京情報クリエイター工学院

ベクトル１

◆ベクトルとは？

問題１「机の上に、リンゴが４個あります。その机に、さらにリンゴを３つ載せました。
　　机の上にリンゴは何個載っていますか？」

解答＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

問題２「北に向かって ４Ｋm 歩きました。その後、東に向かって ３Ｋm 歩きました。
　　今、出発点からどれだけ離れた所に居ますか？」

解答＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

上記の問題の違いは何かわかりますか？
問題２は、「４＋３」ではありませんね。「歩く」には「向き」がありますが、「リンゴの個数」には「向き」がありません。
「歩く」事に限りません。世の中には「向きを持った量」はいっぱいあります。中学生の理科で学んだ「速度」・「加速度」・「力」はもちろん、「性格」「考え方」などにも向きがあります。

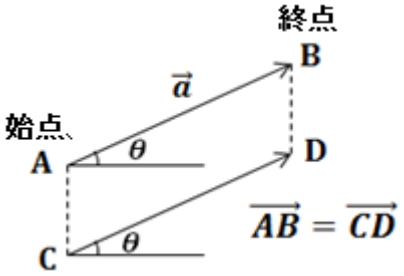
このように「向きのある量」は、単純に数字で表す事はできません。そこでそのような「向きを持った量」は「矢印」を使って表します。こうすれば、さっきの「北に４Ｋm歩いて・・・」という問題も、きちんと「５Ｋm」という結論が出せるでしょう。

矢印で表さなくてならない「**大きさと向き**」をもつものを「　**ベクトル**　」といいます。「歩く」のは「　**ベクトル**　」です。これに対して「リンゴの数」みたいに、「**大きさだけ**」で方向の無い量を「　**スカラー**　」と言います。

ベクトルとは
ある空間での「　　」と「　　」を持ち、「　　」ものである。

◆ベクトルの表し方

ベクトルは、「**線分の長さ**」で**ベクトルの**「　**大きさ**　」を、「**矢印の向き**」で**ベクトルの**「　**向き**　」を表します。



そして、図のように、点 **A** から点 **B** に向かう有向線分（向きのある線分）で表されるとき、このベクトルを「　　」と書き、**A** を「　**始点**　」 **B** を「　**終点**　」と呼びます。

ここで、**有向線分を平面上のどの点を始点として書くかは問題にしません**。「ベクトルとは」のところでも「　**位置情報を持たない**　」とありましたね。

つまり、平面上で、ベクトル $\overrightarrow{AB}$ の代わりに向きと大きさが等しいベクトル $\overrightarrow{CD}$ を図のように書いても、2 つのベクトル $\overrightarrow{AB}$ と $\overrightarrow{CD}$ は**同じである**と考え、「　　」で表します。
このため、有向線分 $\overrightarrow{AB}$ が表すベクトルを 1 つの文字 $\vec{a}$ と表すことがあります。

「　ベクトル１　」

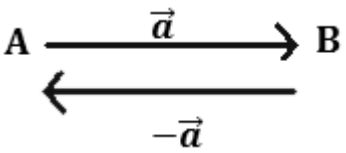
東京情報クリエイター工学院

また、ベクトル $\overrightarrow{AB}$ や $\vec{a}$ の大きさは、絶対値の記号を用いて、「　　」のように表します。ベクトルの大きさは、「　**ノルム**　」ともいいます。

◆いろいろなベクトル

大きさが「　1　」で**任意の向き**を持つベクトルを「　　」**ベクトル**といい、「　　」で表します。また、始点 **A** と終点 **B** が一致するベクトル（**AB** は **AA** となるベクトル）を大きさが零のベクトルと考え、「　　」**ベクトル**といいます。零ベクトルは、**任意の向き**を持ち、「　　」で表します。

そして、$\vec{a}$ と大きさが等しく、向きが反対であるベクトルを「　　」**ベクトル**といい、「　　」で表します。



【逆ベクトルと零ベクトル】

$$\vec{a} + (-\vec{a}) = \vec{0}$$
$$\vec{a} + \vec{0} = \vec{a}$$

【例題１】

マリオブラザーズの簡易版で遊んでいます。簡易版では、マリオは左右方向にしか動けません(ジャンプもしない)。マリオは、水平方向 200 ピクセルの位置をスタートして、右に動きます。ところが、250 ピクセルの位置でキノコを採り忘れたことを思い出し、キノコを採りに 100 ピクセルの位置に戻ります。キノコを採った後、マリオは 450 ピクセルの位置にいるプリンセスのもとに走っていきます。マリオの移動した変位と実際に走った距離を求めなさい。

まず変位を計算します。マリオは 200 の位置からからスタートし、450 ピクセルの位置にたどり着いたので、

変位＝最終位置－最初の位置＝

距離を計算するときは、向きは気にしなくてよいので、マリオが走った長さを足し合わせます。
まず、スタート地点である 200 ピクセルの位置から 250 ピクセルの位置までの長さは

」

そこからキノコを採りに 100 ピクセルに戻るから

」

ここから、プリンセスの位置 450 ピクセルに行くので

」

これらをすべて加えて

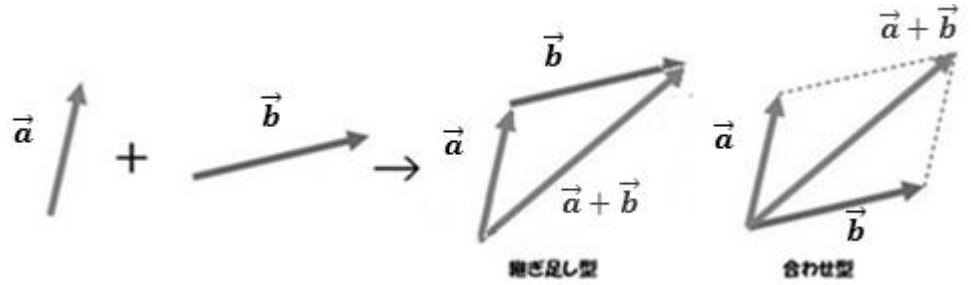
マリオの例からわかるように、距離と変位とは全く異なる値になることに注意しましょう。また、変位を考えるときに、途中の経路で起きた情報は、一切無視されることにも注意しましょう。どんな情報も見落としたくない場合は、運動を短い時間間隔に分割すればよいことになります。例えば、マリオの場合、シナリオは 3 つのセグメントに分けられます。プログラムを書くときの注意点として、向きを考慮するときには必ず変位を使わなければならないことを覚えておきましょう。従って、マリオの場合、＋50 ピクセル、キノコまでの－150 ピクセル、そしてプリンセスまでの＋350 ピクセルという 3 つのセグメントで考えることになります。

「　ベクトル１　」

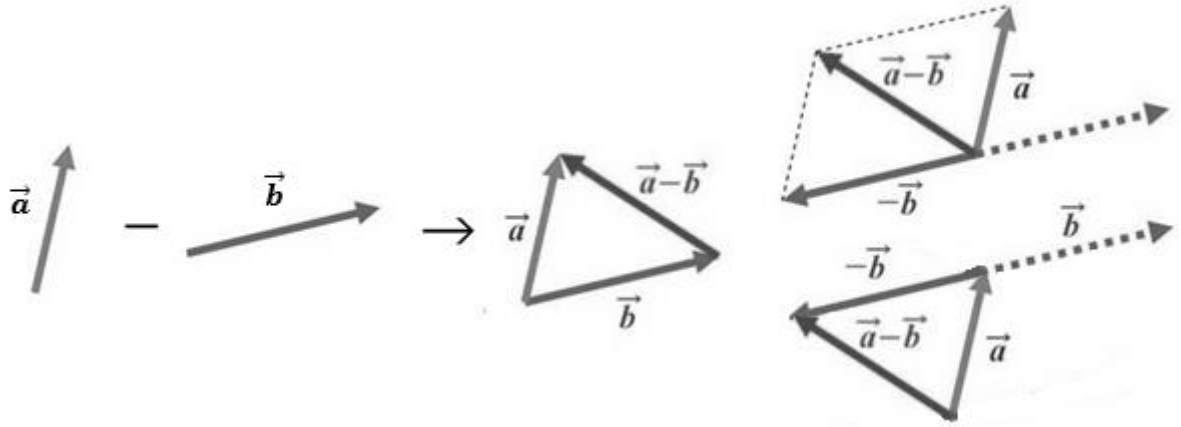
東京情報クリエイター工学院

◆ベクトルの加減算

１．加算

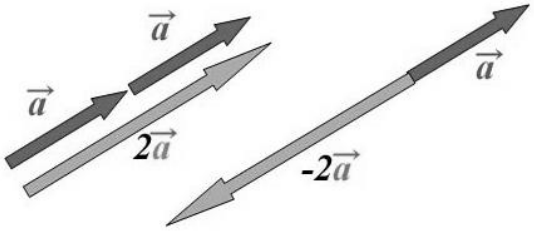


２．減算



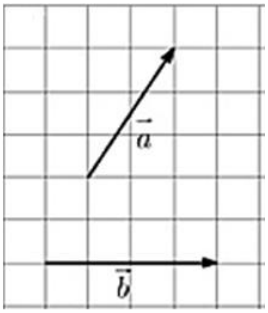
３．スカラー倍

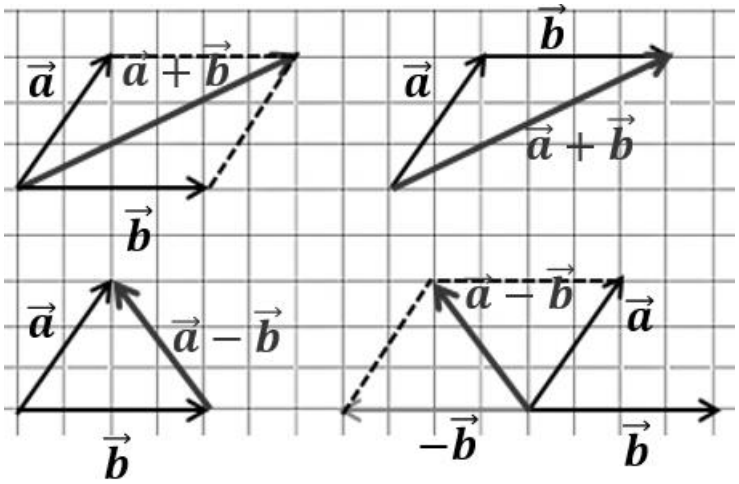
ベクトルのスカラー倍は、「方向を変えずに大きさをスカラー倍」すること。マイナス倍は、「方向が逆」になる。



【例題１】

$\vec{a}\,\vec{b}$ が次のように与えられているとき、$\vec{a}+\vec{b}$ $\vec{a}-\vec{b}$　をそれぞれ図示しなさい。



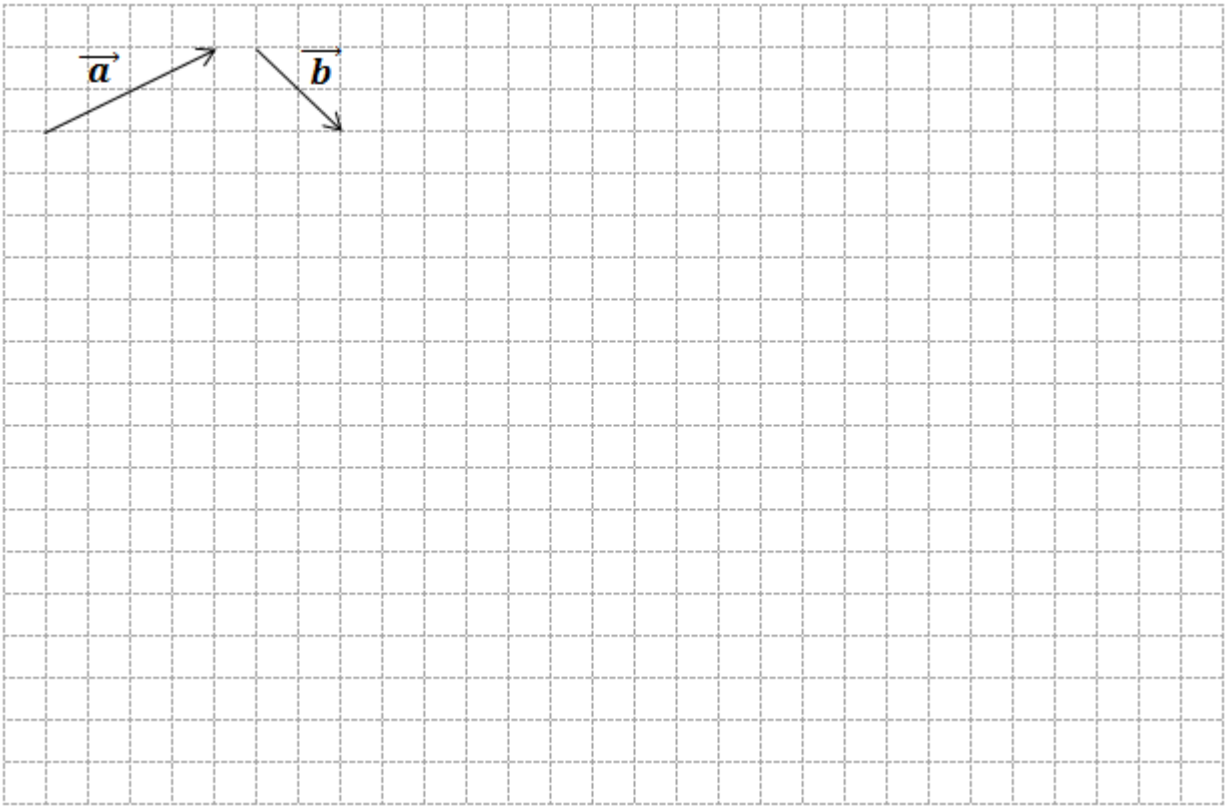


「　ベクトル１　」

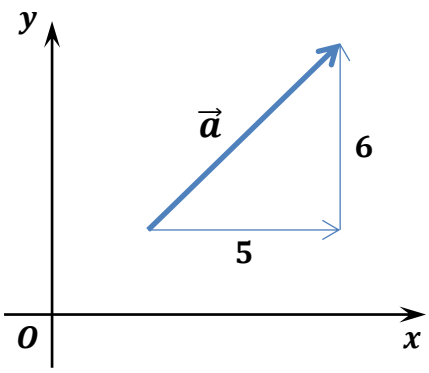
東京情報クリエイター工学院

【演習問題１】

$\vec{a}\,\vec{b}$ が次のように与えられているとき、$\vec{a}+\vec{b}$　$2\vec{a}-3\vec{b}$　をそれぞれ図示しなさい。



◆ベクトルの成分



ベクトルの成分表記については、それぞれ水平・垂直方向の成分に分けて表記し、演算に関しては、成分ごとに演算することになります。

図のように示されるベクトル $\vec{a}$ は、水平方向を x 成分、垂直方向を y 成分といい、次のように表します。

「　$\vec{a}=$　　」

ここで、ベクトルは位置の情報を持たないので、座標上どこにあっても構いませんね。

言い換えると矢印の始点を決めれば、ベクトル位置が決まるので、「ある点をどの方向にどのくらい移動するのか」という考えをもってベクトルは扱うと理解しやすいですね。ゲームではキャラクタが現在位置から次のフレームにどの方向へどのくらいの距離進むかを考えるとき使用したりします。

「　ベクトル１　」

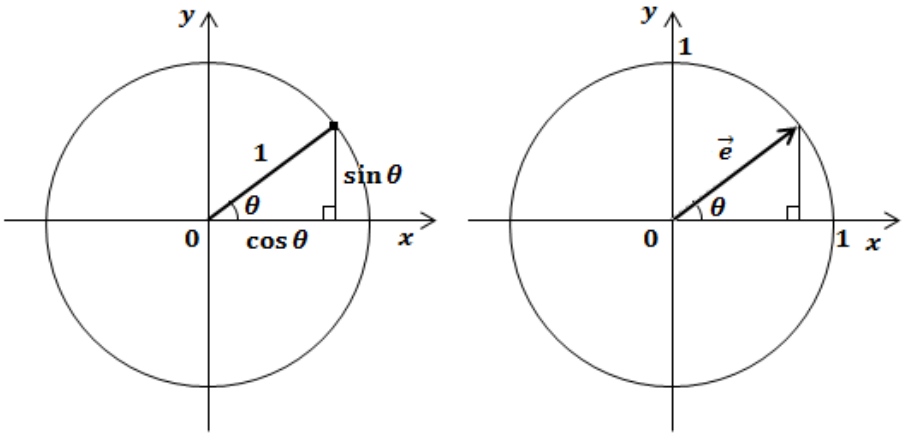
東京情報クリエイター工学院

◆ベクトルと三角関数

三角関数とベクトルについて考えてみましょう。
半径「1」の円「単位円」において、半径「1」ですから、左図では、

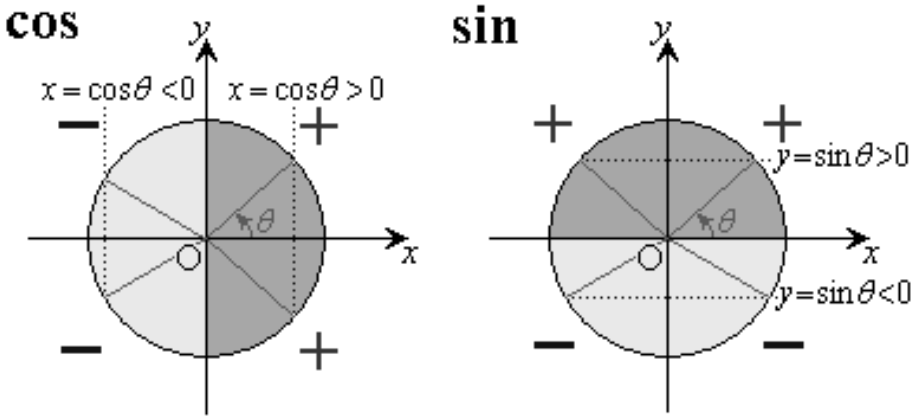
$$x = \cos\theta \qquad y = \sin\theta$$

となります。右の図も単位円です。そこで、なす角 θ の $\vec{e}$ を考えます。
単位円ですから、「 $|\vec{e}| = 1$ 」そして、その成分は、$\vec{e} = (\cos\theta, \sin\theta)$ となります。
このとき、$\vec{e}$ を「　　」**ベクトル**といいます。



【単位ベクトル】

単位ベクトルの x 成分　　$\cos\theta$
単位ベクトルの y 成分　　$\sin\theta$
単位ベクトル　　$\vec{e} = (\cos\theta, \sin\theta)$



「　ベクトル１　」

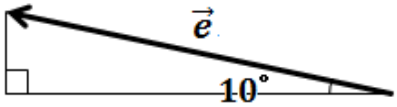
東京情報クリエイター工学院

【例題２】

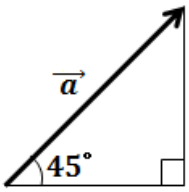
次のベクトルの x 成分 y 成分を求めなさい。

ただし、$\cos 10° = 0.98$ $\sin 10° = 0.17$ $\cos 45° = \sin 45° = 0.71$ $|\vec{a}| = 10$ とします。

(1)



(2)

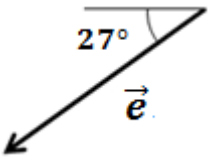


【演習問題２】

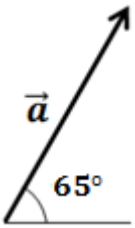
次のベクトルの x 成分 y 成分を求めなさい。

ただし、$\cos 27° = 0.89$ $\sin 27° = 0.45$ $\cos 65° = 0.42$ $\sin 65° = 0.91$ $|\vec{a}| = 5$ とします。

(1)



(2)



「　ベクトル１　」

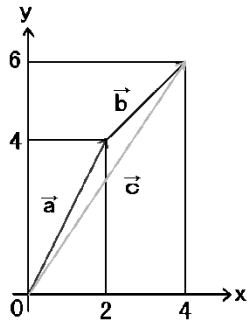
東京情報クリエイター工学院

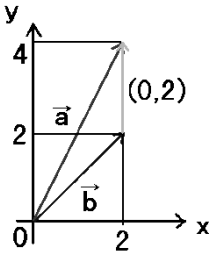
◆成分による演算

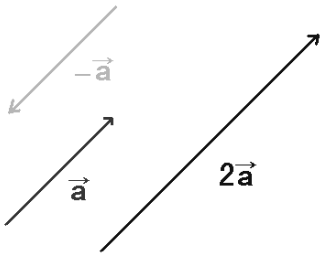
ベクトル $\vec{a}=(2,4)$ $\vec{b}=(2,2)$ を成分で演算する場合、**成分どうしを足せば良い**のです。引き算も同じです。

$$\vec{a}+\vec{b}=(2,4)+(2,2)=(4,6) \qquad \vec{a}-\vec{b}=(2,4)-(2,2)=(0,2)$$
$$2\vec{a}=2\times(2,4)=(4,8)$$

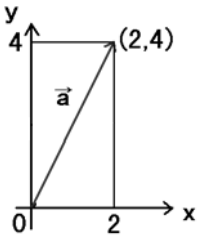
次に、ベクトルに実数を掛ける場合ですが、例えば $\vec{a}$ に「 2 」を掛ける場合を考えましょう。$2\times\vec{a}(=2\vec{a})$ は、$\vec{a}$ の方向を変えずに、大きさを 2 倍にすると約束します。当然 $3\vec{a}$ ならば、大きさを 3 倍にしたものです。マイナス倍は「逆向き」と定義します。$-\vec{a}$ は、$\vec{a}$ の逆向きですね。従って、$-2\vec{a}$ は逆向きで大きさが 2 倍ということです。まとめると、$\vec{a}$ の成分両方に実数を掛けるということです。







◆ベクトルの大きさ



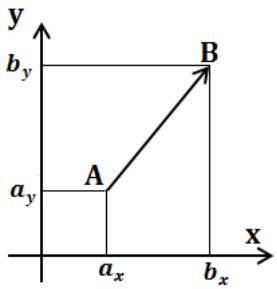
ベクトルの大きさを求めるには、$\vec{a}=(x,y)$ のとき、

「　$|\vec{a}|=$　　」

となります。従って、$\vec{a}=(2,4)$ のときは、　$|\vec{a}|=$

また、図のように、２つの点 $\mathrm{A}(a_x,a_y)$　$B(b_x,b_y)$ に対して $\overrightarrow{AB}$ は

$$\overrightarrow{AB}=(b_x-a_x,b_y-a_y)$$



で表すことができますから、 $\overrightarrow{AB}$ の大きさは

$$\left|\overrightarrow{AB}\right|=$$

例えば、$\mathrm{A}(2,4)$　$B(3,6)$ のとき、$\overrightarrow{AB}$と $\left|\overrightarrow{AB}\right|$ を求めてみましょう。

$$\overrightarrow{AB}=$$

$$\left|\overrightarrow{AB}\right|=$$

「　ベクトル１　」

東京情報クリエイター工学院

【成分による演算】

$\vec{a}=(a_x, a_y)$　$\vec{b}=(b_x, b_y)$ とするとき、

$\vec{a}+\vec{b}=(a_x+b_x,\ a_y+b_y)$　$\vec{a}-\vec{b}=(a_x-b_x,\ a_y-b_y)$

$k\vec{a}=\ \vec{a}k=(ka_x,\ ka_y)$

$|\vec{a}|=\sqrt{{a_x}^2+{a_y}^2}$　　$|\overrightarrow{ab}|=\sqrt{(b_x-a_x)^2+(b_y-a_y)^2}$

【ベクトルの演算法則】

交換法則　$\vec{a}+\vec{b}=\vec{b}+\vec{a}$　　**結合法則**　$(\vec{a}+\vec{b})+\vec{c}=\vec{a}+(\vec{b}+\vec{c})$

分配法則　$k(\vec{a}+\vec{b})=k\vec{a}+k\vec{b}$　$(k+l)\vec{a}=k\vec{a}+l\vec{a}$　$k(l\vec{a})=(kl)\vec{a}=kl\vec{a}$

◆演習問題3

１．$\vec{a}=(1,-2)$　$\vec{b}=(-1,-3)$ であるとき、次の計算をしなさい。

⑴　$\vec{a}+\vec{b}=$

⑵　$\vec{a}-\vec{b}=$

⑶　$4\vec{a}=$

⑷　$2\vec{a}+3\vec{b}=$

⑸　$2\vec{a}-3\vec{b}=$

⑹　$|\vec{a}|=$

２．2 点 $A(3,-1)$　$B(-1,-4)$ において、$\overrightarrow{AB}$　$|\overrightarrow{AB}|$ を求めなさい。

３．4 点 $A(1,5)$　$B(-2,1)$　$C(0,-1)$　D を頂点とする平行四辺形 $ABCD$ があります。頂点 D の座標を求めなさい。

「　ベクトル１　」

東京情報クリエイター工学院

◆ベクトルのプログラム1

１．実習で使用するライブラリについて

実習で使用するライブラリについて次の２つを使用します。

①「MathUtility.dll」　ベクトル、行列、角度の単位変換など

②「DrawUtility.dll」　描画に関する処理

①に関しては別途各自で作成していきますが、必要に応じて使用します。

②はフォームアプリで簡易に描画ができるように工夫したライブラリです。

ベクトルや行列計算においてその計算結果を図形描画して確認するために面倒な描画処理をライブラリ化しています。

２．ベクトルクラスライブラリの作成

２次元ベクトルの計算を理解するために、2 次元ベクトルのライブラリを作成しましょう。

(1)[V_学籍番号_氏名]フォルダをコピーし、[MathCalc]ソリューションを開く。

(2)[MyLibrary]－[Vector2.cs]を開く。

(3)ベクトル型の定義。以下を入力。

```
namespace MyLibrary
{
    //ベクトルクラスの定義
    public struct Vector2
    {
        //ベクトルの要素
        public float X;
        public float Y;

        // </summary>
        // ベクトルの値を代入する。
        // <param name="x">X 座標値</param>
        // <param name="y">Y 座標値</param>
        public Vector2(float x, float y)
        {
            this.X = x;
            this.Y = y;
        }

        // <summary>
        // ゼロベクトルを返します。
        // </summary>
        public static Vector2 Zero
        {
            get
            {
                return new Vector2(0.0f, 0.0f);
            }
        }

        //ベクトル表記の文字列　(a,b)の形
        public override string ToString()
        {
            return "(" + X + "," + Y + ")";
        }
```

```
        //暗黙の型変換をする　配列データの取得時に使用するため
        public static implicit operator float[] (Vector2 p)
        {
            float[] result = new float[] { p.X, p.Y };
            return result;
        }

        //これ以降に、ベクトルの計算・処理を追加していきます。

    }

}
```

(5)ベクトルの加算

ベクトルの加算を求めます。　**引数：２つのベクトル　　戻り値：ベクトル**

ベクトルの加算は　それぞれの成分ごとに和を求めればいいので、

ベクトルが　$\vec{a}(a_x, a_y)$ $\vec{b}(b_x, b_y)$ のとき　$\vec{a}+\vec{b}=(a_x+b_x\ ,\ a_y+b_y)$　となりますね。

```
// <summary>
// ベクトルの加算
// ベクトル１とベクトル２を加算します。
// </summary>
// <param name="v1">ベクトル１</param>
// <param name="v2">ベクトル２</param>
// <returns>ベクトルの和</returns>
public static Vector2 Add(Vector2 vector1, Vector2 vector2)
{
    vector1.X += vector2.X;
    vector1.Y += vector2.Y;
    return vector1;
}
```

(6)ベクトルの加算を参考に、以下の処理を加える。

①ベクトルの減算：Subtract(Vector2 vector1, Vector2 vector2)
引数：２つのベクトル　　戻り値：ベクトル

②ベクトルのスカラー倍：Multiply(Vector2 vector, float scalar)
引数：ベクトル、スカラー　　戻り値：ベクトル

③ベクトルの乗算(各成分同士を乗算する)：Multiply(Vector2 vector1, Vector2 vector2)
引数：2 つのベクトル　　戻り値：ベクトル

④ベクトルをスカラーで割り算：Divide(Vector2 vector, float scalar)
引数：ベクトル、スカラー　　戻り値：ベクトル

⑤ベクトルの除算(ベクトル 1 をベクトル 2 で除算)
：Multiply(Vector2 vector1, Vector2 vector2)
引数：2 つのベクトル　　戻り値：ベクトル

⑥ベクトルの大きさ：Length()と Length(Vector2　vector)
引数：ベクトル　　戻り値：スカラー
使用方法：a ベクトルの大きさであれば、Vector2.Length(a)

⑦ベクトルの大きさの 2 乗：LengthSquared()と LengthSquared(Vector2 vector)
引数：ベクトル　　戻り値：スカラー

(7)演算子の使用

演算子（+,-,*,/）を使用できるようにする。

```
// 符号として　単項演算子＋のオーバーロード
// （ほとんど必要ないけど、使うこともあるので）
public static Vector2 operator +(Vector2 vector)
{
    return vector;
}

// 符号として　単項演算子－のオーバーロード。
// （ベクトルの方向を変えることができます）
public static Vector2 operator -(Vector2 vector)
{
    vector *= -1;
    return vector;
}

// ベクトルの加算　＋演算子をオーバーロード
public static Vector2 operator +(Vector2 vector1, Vector2 vector2)
{
    return Add(vector1, vector2);
}

// ベクトルの減算　－演算子をオーバーロード
public static Vector2 operator -(Vector2 vector1, Vector2 vector2)
{
    return Subtract(vector1, vector2);
}

//ベクトルのスカラー倍①　*演算子をオーバーロード
// スカラー×ベクトル
public static Vector2 operator *(float scalar, Vector2 vector)
{
    return Multiply(vector, scalar);
}

// ベクトルのスカラー倍②　*演算子をオーバーロード
// ベクトル×スカラー
public static Vector2 operator *(Vector2 vector, float scalar)
{
    return Multiply(vector, scalar);
}

//ベクトル同士の掛け算
public static Vector2 operator *(Vector2 vector1, Vector2 vector2)
{
    return Multiply(vector1, vector2);
}

// ベクトルをスカラーで除算します。
public static Vector2 operator /(Vector2 vector, float scalar)
{
    return Divide(vector, scalar);
}

// ベクトル１をベクトル２で除算します。
public static Vector2 operator /(Vector2 vector1, Vector2 vector2)
{
    return Divide(vector1, vector2);
}
```

「　ベクトル１　」

東京情報クリエイター工学院

３．ライブラリの使用

作成したライブラリの計算が正しいかどうか以下の計算をコンソールアプリで実行し、確認しましょう。

$\vec{a} = (6,3)\ \vec{b} = (-3,1)\ \vec{c} = (3,5)$ として、

$\vec{a} + \vec{b} = (3,4)\ \vec{b} - \vec{c} = (-6,-4)\ \ 3 \times \vec{c} = (9,15)\ \ |\vec{a} + \vec{b}| = 5$

①[MathCalc]ソリューション内にコンソールアプリケーションを作成する。

プロジェクト名　[Ex01]

②参照の追加

[MyLibrary]を参照設定に追加する。

③using MyLibrary;　　を忘れずに！

④[Ex01]をスタートアップ プロジェクトに設定

【Vector2.cs　プログラム例】

```
        // <summary>
        // ベクトルの減算
        // ベクトル１からベクトル２を減算します。
        // </summary>
        // <param name="v1">ベクトル１</param>
        // <param name="v2">ベクトル２</param>
        // <returns>減算されたベクトル</returns>
        public static Vector2 Subtract(Vector2 vector1, Vector2 vector2)
        {
            vector1.X -= vector2.X;
            vector1.Y -= vector2.Y;
            return vector1;
        }

        // <summary>
        // ベクトルとスカラー値を乗算します。
        // （ベクトルの長さを変えるときに使用します）
        // </summary>
        // <param name="v1">ベクトル</param>
        // <param name="scalefactor">スケール（倍率）</param>
        // <returns></returns>
        public static Vector2 Multiply(Vector2 vector, float scale)
        {
            vector.X *= scale;
            vector.Y *= scale;
            return vector;
        }

        // <summary>
        // ベクトルの乗算（各成分を乗算します）
        // </summary>
        // <param name="vector1">ベクトル１</param>
        // <param name="vector2">ベクトル２</param>
        // <returns></returns>
        public static Vector2 Multiply(Vector2 vector1, Vector2 vector2)
        {
            vector1.X *= vector2.X;
            vector1.Y *= vector2.Y;
            return vector1;
        }

        // <summary>
        // ベクトルをスカラー値で除算します。
        // （ベクトルの長さを変えるときに使用します）
        // </summary>
        // <param name="vector">ベクトル</param>
        // <param name="scalar">スカラー</param>
        // <returns></returns>
        public static Vector2 Divide(Vector2 vector, float scalar)
        {
            if (scalar == 0) return Zero;
            vector.X /= scalar;
            vector.Y /= scalar;
            return vector;
        }
```

```
        // <summary>
        // ベクトルの除算　（各成分同士で割る）
        // </summary>
        // <param name="vector1">ベクトル１</param>
        // <param name="vector2">ベクトル２</param>
        // <returns></returns>
        public static Vector2 Divide(Vector2 vector1, Vector2 vector2)
        {
            if (vector2.X == 0 || vector2.Y == 0) return Zero;
            vector1.X /= vector2.X;
            vector1.Y /= vector2.Y;
            return vector1;
        }

        // <summary>
        // ベクトルの大きさ（長さ）を返します。
        // </summary>
        public float Length()
        {
            return (float)Math.Sqrt(LengthSquared(this));
        }

        // <summary>
        // ベクトルの大きさ（長さ）を返します。
        // </summary>
        public static float Length(Vector2 vector)
        {
            return (float)Math.Sqrt(LengthSquared(vector));
        }

        // <summary>
        // ベクトルの大きさの平方（２乗）を返します。
        // </summary>
        public float LengthSquared()
        {
            return X * X + Y * Y;
        }

        // <summary>
        // ベクトルの大きさの平方（２乗）を返します。
        // </summary>
        public static float LengthSquared(Vector2 vector)
        {
            return vector.X * vector.X + vector.Y * vector.Y;
        }
```

「　ベクトル１　」

東京情報クリエイター工学院

【Ex01 Program.cs　プログラム例】

```
using MyLibrary;

namespace Ex01
{
    class Program
    {
        static void Main(string[] args)
        {
            Vector2 a = new Vector2(6, 3);
            Vector2 b = new Vector2(-3, 1);
            Vector2 c = new Vector2(3, 5);

            Vector2 ans1 = a + b;
            Console.WriteLine("a+b=" + ans1);

            Vector2 ans2 = b - c;
            Console.WriteLine("b-c=" + ans2);

            Vector2 ans3 = 3.0f * c;
            Console.WriteLine("3*c=" + ans3);

            float ans4 = Vector2.Length(ans1);
            Console.WriteLine("a+b の大きさ：" + ans4);
        }
    }
}
```