「　ベクトル２　」

東京情報クリエイター工学院

ベクトル２

◆ベクトルの正規化

$\vec{a}$ と同じ方向の単位ベクトルを求める演算を $\vec{a}$の「　**正規化**(normalize)　」といいます。

$\vec{a} = (a_x, a_y)$ の正規化

$$normalize(\vec{a}) = \frac{\vec{a}}{|\vec{a}|} = \frac{1}{|\vec{a}|}\vec{a} = \frac{1}{\sqrt{{a_x}^2 + {a_y}^2}}(a_x, a_y) = \left(\frac{a_x}{\sqrt{{a_x}^2 + {a_y}^2}}, \frac{a_x}{\sqrt{{a_x}^2 + {a_y}^2}}\right)$$

【例題１】

次のベクトルを正規化しなさい。

$\vec{a} = (3, 4)$

$|\vec{a}| = \sqrt{{a_x}^2 + {a_y}^2} =$

$normalize(\vec{a}) = \frac{1}{|\vec{a}|}\vec{a} =$

【練習問題１】

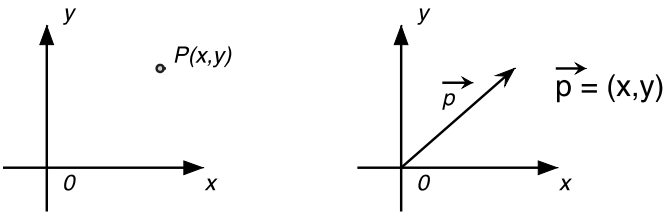
次のベクトルを正規化しなさい。

① $\vec{a} = (-30, 40)$

② $\vec{b} = (12, -5)$

◆位置ベクトル

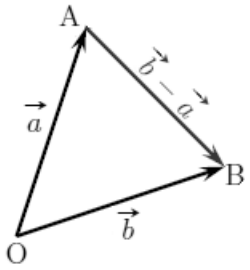
「　**位置**　」**ベクトル**とは、**原点を始点とするベクトル**のことです。原点を始点に決めれば、点の位置をベクトルで表すことができます。



任意の点 P (x, y) の位置ベクトルは、$\vec{p} = (x, y)$ となります。
逆に、点 P の位置ベクトル $\vec{p}$ が、(x, y) であるとき、点 P の座標は、(x, y) となります。
つまり、点 P の座標値と位置ベクトル $\vec{p}$ の成分は同じ値をとります。

「　ベクトル２　」

東京情報クリエイター工学院



任意の 2 点 $A\ (a_x, a_y)$ $B\ (b_x, b_y)$ に対して、位置ベクトルは、

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

点 A を始点、点 B を終点とするベクトル $\overrightarrow{AB}$ は

「 $\overrightarrow{AB} = \vec{b} - \vec{a} = (b_x - a_x, b_y - a_y)$ 」

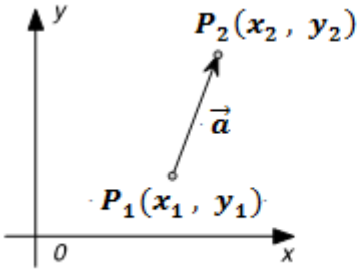
となります。

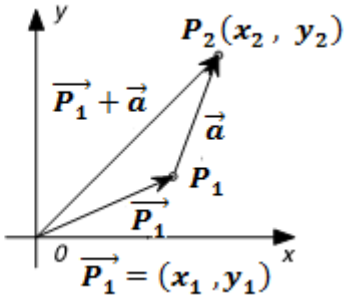
また、任意の点 $P_1(x_1,\ y_1)$ と $\vec{a} = (a_x, a_y)$ において、

$$P_2(x_2,\ y_2) = P_1 + \vec{a} = (x_1 + a_x,\ y_1 + a_y)$$

$$\vec{a} = P_2 - P_1 = (x_2 - x_1,\ y_2 - y_1)$$

すなわち、座標点とベクトルを加えると、座標点が得られます。(厳密には座標点ではなく位置ベクトルなので、単純にベクトルの演算をしているにすぎません)。





【練習問題２】

(1) 座標点 $P_1(40, 30)$、$P_2(50,\ 20)$ のとき、P_1 を始点とする P_2 を終点とするベクトル $\overrightarrow{P_1P_2}$ を求めなさい。

(2) ベクトル $\vec{a} = (10, 5)$ の始点が、 座標点 $P_1(30, 40)$ にあるとき、終点 P_2 の座標値を求めなさい。

(3) 次の計算をしなさい。ただし、ベクトル・スカラー・座標点は以下の値とします。

$\vec{a} = (2, 3)$ $\vec{b} = (5, 2)$ $\vec{c} = (1, 2)$ $L = 3$ $M = 2$ $P_1(5, 3)$ $P_2(4, 1)$

① $\vec{a}L$　　**② $M\vec{b}$**

③ $P_1 + \vec{b}$　　**④ $\vec{a}M + \vec{b}$**

⑤ $(L + M)\vec{c}$　　**⑥ $P_1 - P_2$**

⑦ $M\vec{c} + L\vec{a}$　　**⑧ $(P_2 - P_1)L$**

「　ベクトル２　」

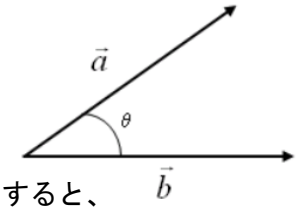
東京情報クリエイター工学院

◆2 次元ベクトルの内積

【内積の定義】

2 つのベクトル $\vec{a}$ $\vec{b}$ 、そのなす角を θ とすると

$$\vec{a}\cdot\vec{b}=|\vec{a}||\vec{b}|\cos\theta$$



また、2 つのベクトルの成分を $\vec{a}=(a_x,a_y)$ $\vec{b}=(b_x,b_y)$ とすると、

$$\vec{a}\cdot\vec{b}=a_xb_x+a_yb_y$$

【例題２】次のベクトルの内積を求めなさい。

(1) $|\vec{a}|=4$ $|\vec{b}|=3$ $\theta=30°$

$\vec{a}\cdot\vec{b}=|\vec{a}||\vec{b}|\cos\theta=$

(2) $\vec{a}=(2,3)$ $\vec{b}=(4,6)$

$\vec{a}\cdot\vec{b}=a_xb_x+a_yb_y=$

【練習問題３】

次のベクトルの内積を求めなさい。

(1) $|\vec{a}|=5$ $|\vec{b}|=6$ $\theta=60°$

(2) $|\vec{a}|=5$ $|\vec{b}|=5$ $\theta=45°$

(3) $\vec{a}=(4,2)$ $\vec{b}=(3,4)$

(4) $\vec{a}=(-1,2)$ $\vec{b}=(2,5)$

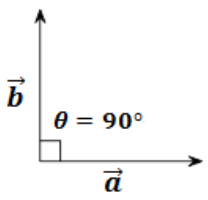
◆内積の利用

１．垂直チェック

垂直チェック

$$\vec{a}\cdot\vec{b}=0 \quad\Leftrightarrow\quad \vec{a}\perp\vec{b}$$

または、 $a_xb_x+a_yb_y=0 \quad\Leftrightarrow\quad \vec{a}\perp\vec{b}$



これを利用すると２つのベクトルが直行しているかを確かめるには、内積の結果が 0 になるかどうかを判断すればよいことになります。直交条件として内積が使えるわけです。

「　ベクトル２　」

東京情報クリエイター工学院

【例１】

$\vec{a}=(-2,2)\ \vec{b}=(4,4)$ のとき、2 つのベクトルが垂直かどうか調べなさい。

【解答】

$\vec{a}\cdot\vec{b}=0$ **ならば、**$\vec{a}$ **と** $\vec{b}$ は垂直ですから、$\vec{a}\cdot\vec{b}$ を求めます。

$\vec{a}\cdot\vec{b}=$

内積の答えが「　　」ですから、2 つのベクトルは

２．なす角を求める

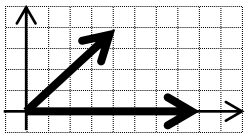
2 つのベクトルのなす角

$\vec{a}$ **と** $\vec{b}$ **を正規化すると、（**$|\vec{a}|=1\ \ |\vec{b}|=1$）

$\vec{a}\cdot\vec{b}=\cos\theta$

【例 2】　$\vec{a}=(4,4)\ \vec{b}=(8,0)$ のとき、2 つのベクトルの角度を求めなさい。

$\vec{a}$ の正規化 $=\dfrac{\vec{a}}{|\vec{a}|}=$



$\vec{b}$ の正規化 $=\dfrac{\vec{b}}{|\vec{b}|}=$

正規化した$\vec{a}$ と $\vec{b}$ の内積を求める。

$\vec{a}\cdot\vec{b}=$

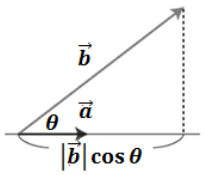
従って、$\theta=45°$

３．正射影されたときの大きさとベクトル

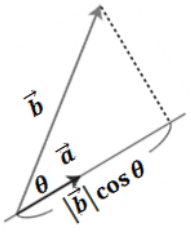
正射影されたときの大きさ

$\vec{a}$　$\vec{b}$ の 2 つのベクトルにおいて、$\vec{a}$ が正規化されているとき

$\vec{a}\cdot\vec{b}=|\vec{a}||\vec{b}|\cos\theta=|\vec{b}|\cos\theta$



また、これは図のように傾きを持っても同じです。斜面がどのように傾いていても、同様に求めることができるところが、ベクトルの便利なところです。

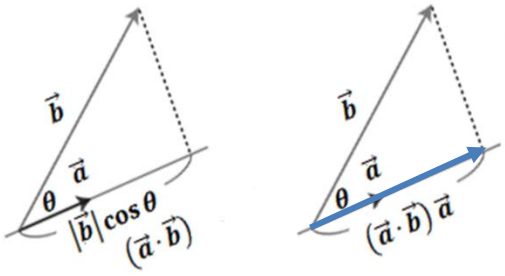


このようにどちらかのベクトルが正規化されているベクトルの内積の値は、

「ベクトル$\vec{b}$**が、正規化された**$\vec{a}$**を含む直線（壁や地面、その他オブジェクト）へ向けて垂直に下ろした（正投影された）時の大きさ（長さ）」**となります。

「　ベクトル２　」

東京情報クリエイター工学院



さらに内積によって求められた長さ（$\vec{a}\cdot\vec{b}$）をその方向である正規化されたベクトル $\vec{a}$ を掛けると（図右）、斜面方向のベクトルが分かります。

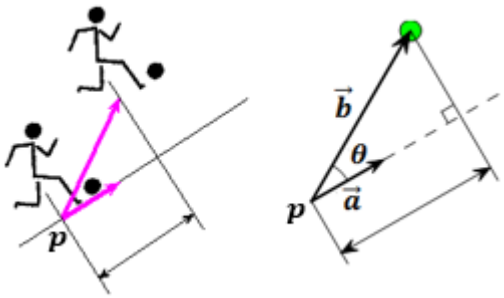
これは、斜面上に置かれたオブジェクトの座標点を簡単に求めることができるということになります。それも三角関数の計算はこの中には入っていないということがポイントです（内積の計算 $\vec{a}\cdot\vec{b} = a_x * b_x + a_y * b_y$ ）。

これがゲームプログラミングにおいての内積のイメージとなり、衝突判定などに多く使用されることになります。これは３次元ベクトルにおいても同様です。

内積は、プログラミングでは、あまり大きな処理パワーを必要としないので、様々なアプリケーションでも使われます。しっかり理解して、使えるようになりましょう。

【例題】

図のようにキャラクタが $\vec{a}$ である斜面を移動しているとします。ある場所 p 点でキャラクタが $\vec{b}$ の方向にジャンプしました。このとき、ベクトルをそれぞれ $\vec{a} = (4,3)$, $\vec{b} = (10,15)$ とすると斜面方向にどれだけキャラクタは進みますか？



ベクトル$\vec{a}$の正規化

$$\frac{\vec{a}}{|\vec{a}|}\ (=\vec{A}) =$$

内積を求める　$L = \vec{A}\cdot\vec{b} = A_x b_x + A_y b_y =$

◆演習問題1

１．$\vec{a} = (4,-3)$ のとき、$\vec{a}$ を正規化しなさい。

２．次のベクトルの内積を求めなさい。なお、$cos70° = 0.34$ とします。

(1)　$|\vec{a}| = \sqrt{2}\ \ |\vec{b}| = 4\ \ \theta = 45°$

(2)　$|\vec{a}| = 10\ \ |\vec{b}| = 5\ \ \theta = 70°$

(3)　$\vec{a} = (-2,5)\ \ \vec{b} = (4,4)$

「　ベクトル２　」

東京情報クリエイター工学院

⑷ $\vec{a} = (3,7)$ $\vec{b} = (2,4)$

３．２つのベクトル $\vec{a} = (-4,4)$ $\vec{b} = (6,6)$ の角度を求めなさい。

４．[Vector2.cs]に正規化・内積のメソッドを追加しましょう。

⑴正規化：Normalize

引数：１つのベクトル　　戻り値：ベクトル

⑵内積の計算：Dot

引数：２つのベクトル　　戻り値：スカラー

５．ライブラリの利用

作成したライブラリが正しい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。

$\vec{a} = (3,4)$ $\vec{b} = (-2,5)$ $\vec{c} = (4,4)$ **として、**

$\vec{a}$ **の正規化** $= (0.6, 0.8)$　$\vec{b} \cdot \vec{c} = 12$

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　[Ex02]

②参照の追加

[MyLibrary]を参照設定に追加する。　using MyLibrary;　を忘れずに！

③[Ex02]をスタートアップ プロジェクトに設定

「　ベクトル２　」

東京情報クリエイター工学院

【Vector2.cs　プログラム例】

```
// <summary>
// ベクトルの正規化をする。
// 単位ベクトル（長さ１のベクトル）にする。
// </summary>
public Vector2 Normalize()
{
    float Len = Length(this);
    if (Len < 0.0001f) return this = Zero;
    return this /= Len;
}

// <summary>
// ベクトルの正規化
// 指定ベクトルから単位ベクトルを求める、結果に返す
// </summary>
// <param name="vector">ベクトル</param>
// <param name="result">戻り値</param>
public static void Normalize(ref Vector2 vector, out Vector2 result)
{
    float Len = Length(vector);
    if (Len < 0.0001f) result = Zero;
    result = vector / Len;
}

// <summary>
// ベクトルの正規化（単位ベクトル）求める。
// </summary>
// <param name="vector">ベクトル</param>
// <returns>単位ベクトル</returns>
public static Vector2 Normalize(Vector2 vector)
{
    float Len = Length(vector);
    if (Len < 0.0001f) return Zero;
    return vector / Len;
}

// <summary>
// 内積を求める。
// </summary>
// <param name="vector1">ベクトル１</param>
// <param name="vector2">ベクトル２</param>
// <returns>内積の値（スカラー）</returns>
public static float Dot(Vector2 vector1, Vector2 vector2)
{
    return vector1.X * vector2.X + vector1.Y * vector2.Y;
}
```

【Ex02　Program.cs プログラム例】

```
using MyLibrary;

namespace Ex02
{
    class Program
    {
        static void Main(string[] args)
        {
            Vector2 a = new Vector2(3, 4);
            Vector2 b = new Vector2(-2, 5);
            Vector2 c = new Vector2(4, 4);

            Vector2 ans1 = Vector2.Normalize(a);
            Console.WriteLine("ベクトルａの正規化：" + ans1);

            float ans2 = Vector2.Dot(b, c);
            Console.WriteLine("b・c = " + ans2);
        }
    }
}
```

「　ベクトル２　」

東京情報クリエイター工学院

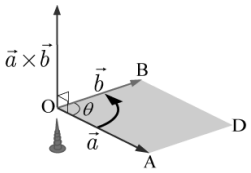
◆2 次元ベクトルの外積

外積は、通常 3 次元空間で定義され、2 つのベクトルを通る平面に対する法線ベクトル(面に垂直なベクトル)が結果として得られます。3 次元の外積については、3 次元ベクトルで説明しますが、概略は、以下の通りです。

【外積の定義(3 次元)】

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

$\vec{a}\times\vec{b}=(a_yb_z-a_zb_y,a_zb_x-a_xb_z,a_xb_y-a_yb_x)$

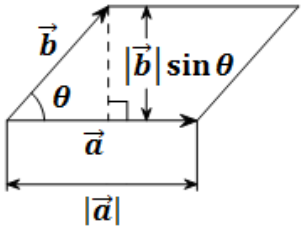


2 つのベクトルに**垂直な方向を向いた**大きさが、$|\vec{a}||\vec{b}|\sin\theta$ のベクトル

【外積の定義(2 次元)】

2 つのベクトル $\vec{a}$　$\vec{b}$ 、そのなす角を θ とすると

$$\vec{a}\times\vec{b}=|\vec{a}||\vec{b}|sin\theta$$



また、2 つのベクトルの成分を $\vec{a}=(a_x,a_y)$　$\vec{b}=(b_x,b_y)$ とすると、

$$\vec{a}\times\vec{b}=(a_xb_y-a_yb_x)$$

2 次元のベクトルの外積の結果は「　**スカラー**　」となり、そしてそれは、$\vec{a}$ **と** $\vec{b}$ **が作る平行四辺形の面積**となる。

【外積の計算方法】

2 次元ベクトル

$\begin{vmatrix}a_x & b_x\\ a_y & b_y\end{vmatrix}$　⇒　**たすき掛け**　$\begin{vmatrix}a_x \overset{\oplus}{\times} b_x\\ a_y \overset{\ominus}{\times} b_y\end{vmatrix}$　⇒　$a_xb_y-a_yb_x$

例えば、$\vec{a}=(4,2)$　$\vec{b}=(3,1)$ のとき

$\begin{vmatrix}4 \overset{\oplus}{\times} 3\\ 2 \overset{\ominus}{\times} 1\end{vmatrix}$　$\vec{a}\times\vec{b}=4\times1-2\times3=4-6=-2$

【練習問題４】

外積$(\vec{a}\times\vec{b})$を計算しなさい。

(1) $\vec{a}=(3,1)$　$\vec{b}=(2,3)$

(2) $\vec{a}=(-2,3)$　$\vec{b}=(4,5)$

「　ベクトル２　」

東京情報クリエイター工学院

◆外積の演算法則

演算法則

(1) 交換法則

$$\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$$　　（注：通常の交換法則は成り立たない）

(2) 分配法則

$$\vec{a} \times \left(\vec{b} + \vec{c}\right) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$$

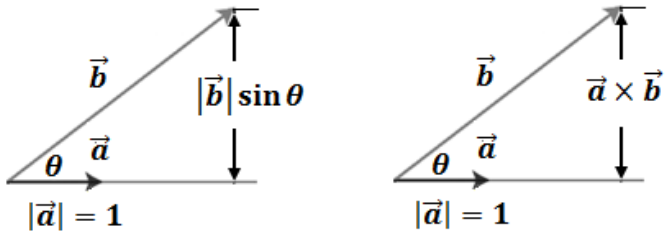
$$\left(\vec{a} + \vec{b}\right) \times \vec{c} = \vec{a} \times \vec{c} + \vec{b} \times \vec{c}$$

(3) 結合法則

$$(\mathrm{m}\vec{a}) \times \vec{b} = m\left(\vec{a} \times \vec{b}\right) = \vec{a} \times (m\vec{b})$$

◆外積の利用

2 次元の外積の利用方法の 1 つが、内積の利用の際に行ったように、1 つのベクトルを正規化（単位ベクトル）したものを利用し、もう 1 つのベクトルを水平方向に投影した大きさ（長さ）がわかるというものです。

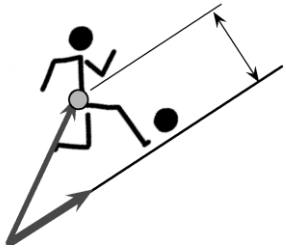


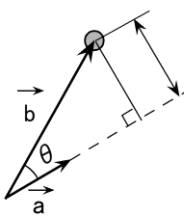
$$|\vec{a}| = 1 \quad \vec{a} \times \vec{b} = |\vec{a}||\vec{b}| \sin\theta = |\vec{b}| \sin\theta$$

これで何が出来るかというと、**ある直線に対して任意点がどのくらいの距離にあるかを知る**ことができます。このことは**キャラクタが斜面に対して垂直に立つ為の位置がわかる**ということです。これはゲームプログラミングをするのには非常に重要で、これが分からないと多分ゲームのプログラムは非常に複雑になってくるほど大切なものです。このイメージがゲームプログラミングにおける 2 次元ベクトルの外積のイメージです。

【例題】

キャラクタがある斜面を移動しています。この斜面に対してキャラクタを垂直に立たせ移動したいので基準座標の位置を斜面からどのくらいの距離にすればいいか求めたい。このとき、ベクトルをそれぞれ$\vec{a} = (4, 3)$, $\vec{b} = (10, 15)$とすると、地面からの距離はどのくらいですか？





ベクトル $\vec{a}$ の正規化

$$\frac{\vec{a}}{|\vec{a}|} (= \vec{A}) =$$

外積を求める

【演習問題１】

「　ベクトル２　」

東京情報クリエイター工学院

１．[Vector2.cs]に外積のメソッドを追加しましょう。

(1)外積の計算：Cross

引数：２つのベクトル　　戻り値：スカラー

２．ライブラリの利用

作成したライブラリが正しい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。

$\vec{a}=(3,4)\ \vec{b}=(-2,5)$ として、$\vec{a}\times\vec{b}=23$

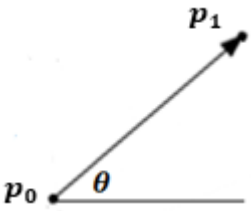
① [MathCalc]ソリューション内のプロジェクト　[Ex02]に追加する。

②参照の追加

[MyLibrary]を参照設定に追加する。　using MyLibrary;　を忘れずに！

③[Ex02]をスタートアップ プロジェクトに設定

３．三角関数１で、マウス入力された座標値と原点を結ぶ線の角度を求めるプログラムを作成しました。それを参考に、マウスから入力された 2 点を始点・終点とするベクトルを描画し、その角度を求める(角度を度で描画)プログラムを作成しましょう。



(0)テンプレートを作る。(テンプレート名：Math03Ex0)

(1)[MathCalc]ソリューション内にテンプレートを使用して、新しいプロジェクトを作成する。(プロジェクト名[Math03Ex01])

(2)[Math03Ex01]をスタートアッププロジェクトに設定する。

【演習問題２】

１．$\vec{a}=(3,-4)$ $\vec{b}=(-1,2)$ 点 $A(3,-1)$ $B(-1,-4)$ であるとき、次の計算をしなさい。

(1) $\vec{a}+\vec{b}=$

(2) $\vec{a}-\vec{b}=$

(3) $2\vec{a}-3\vec{b}=$

(4) $\overrightarrow{AB}=$

(5) $\left|\overrightarrow{AB}\right|=$

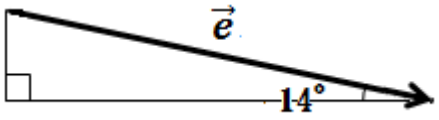
「　ベクトル２　」

東京情報クリエイター工学院

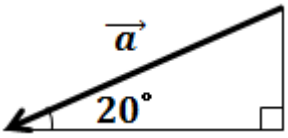
２．次のベクトルの x 成分 y 成分を求めなさい。

ただし、$sin14° = 0.24$　$cos14° = 0.97$　$sin20° = 0.34$　$cos20° = 0.93$　$|\vec{a}| = 20$ とします。

(1)



(2)



３．次のベクトルの内積を求めなさい。なお、$cos70° = 0.34$ とします。

① $|a| = 10$　$|b| = 5$　$\theta = 70°$

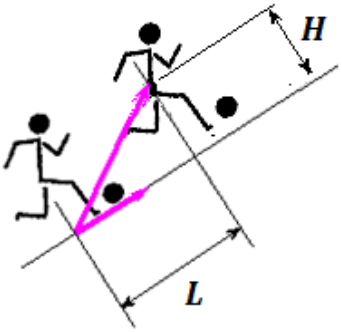
② $\vec{a} = (-2, 5)$　$\vec{b} = (4, 4)$

４．$\vec{a} = (6, 3)$　$\vec{b} = (-3, 1)$ であるとき、

①$\vec{b}$ を正規化しなさい。

②外積を求めなさい。

５．図のようにキャラクタが $\vec{a}$ である斜面を移動しているとします。ある場所 p 点でキャラクタが $\vec{b}$ の方向にジャンプしました。このとき、ベクトルをそれぞれ $\vec{a} = (12, 5)$, $\vec{b} = (13, 13)$ とするとキャラクタは斜面に対して、どの位置にいますか？　L と H を求めなさい。



「　ベクトル２　」

東京情報クリエイター工学院

【Vector2.cs　プログラム例】

```
//Vector2.cs に追加
      // <summary>
      // ２次元での外積　（注：戻り値はスカラー　、Xna にはありません）
      // </summary>
      // <param name="vectorA">ベクトル A</param>
      // <param name="vectorB">ベクトル B</param>
      // <returns>スカラー</returns>
      public static float Cross(Vector2 vector1, Vector2 vector2)
      {
          return vector1.X * vector2.Y - vector1.Y * vector2.X;
      }
//Program.cs に追加
          float ans3 = Vector2.Cross(a, b);
          Console.WriteLine("a×b = " + ans3);
```

【Math03Ex01 MyDrawClass.cs プログラム例】

```
using DrawUtility;
using System.Drawing;  //PointF 型や色設定などに使用
using MyLibrary;

namespace Math03Ex01
{
    public class MyDrawClass : Draws, IDraws
    {
        //フィールドは通常と同じで必要に応じて記述
        InputState input;
        Vector2 p0, p1, p2;   //★修正
        float angle;          //★追加

        //コンストラクタは通常と同じように初期設定などを記述すればいい。
        public MyDrawClass()
        {
            //変更なし
        }

        //必要に応じてその他メソッドを追加する
        public void InputData()
        {
            Init();
            Clear();
            List<PointF> p = new List<PointF>();
            p = input.GetPoint(2, "座標点を２つ入力");  //★修正
            if (p.Count == 0) { Clear(); return; }
            p0 = new Vector2(p[0].X, p[0].Y);

            //★以下追加
            p1 = new Vector2(p[1].X, p[1].Y);

            //角度を求める
            p2 = p1 - p0;
            angle = (float)Math.Atan(p2.Y / p2.X);  //角度を求める
            angle *= 180.0f / (float)Math.PI;       //ラジアンを度に変換

            //3 時方向を 0 度とした角度
            if (p2.X < 0) angle += 180;
            else if (angle < 0) angle += 360;

            Draw();
            Render();
        }

        //Update・Draw メソッドは Animation 実行時に自動更新される
        public void Update()    //主に計算の更新処理を記述
        {
        }

        public void Draw()      //主に描画処理を記述
        {
            Text(p0, angle.ToString());
            Line(p0, p1);
            Dot(p0);
            Dot(p1);
        }
    }
}
```