co-1. クラスとメソッド

（C++ オブジェクト指向プログラミング入門）（全３回）

URL: https://www.kkaneko.jp/pro/cpp/index.html

金子邦彦

アウトライン

- クラス定義
- メソッドの記述
- プログラムの実行

クラス定義

クラス定義の中には，**属性**の定義（**属性名**と**データ型**），**コンストラクタ**の定義，**デストラクタ**の定義，その他**メソッド**の定義を含める．

```
#pragma once
class Ball {
  private:
    double x, y;
  public:
    Ball( const double x, const double y );
    Ball( const Ball& ball );
    Ball& operator= ( const Ball& ball );
    ~Ball();
};
```

属性（メンバ変数ともいう）

コンストラクタ，
デストラクタ

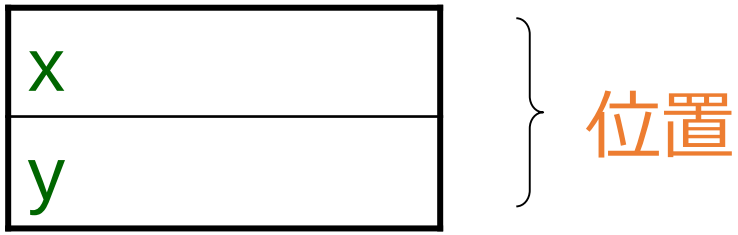
```
#include "Ball.h"
Ball::Ball( const double x, const double y ) : x( x ), y( y )
{
    /* do nothing */
}
Ball::Ball( const Ball& ball ) : x( ball.x ), y( ball.y )
{
    /* do nothing */
}
Ball& Ball::operator= (const Ball& ball )
{
    this->x = ball.x;
    this->y = ball.y;
    return *this;
}
Ball::~Ball()
{
    /* do nothing */
}
```

例題 1

- クラス定義
- クラス定義には，属性定義，コンストラクタ，デストラクタを含める

Ball クラス

- Ball クラス
- Ball は，x, y という 2 つの属性から構成する
- クラス定義には，キーワード class を使用



ソースコード

ファイル名: Ball.h

```
#pragma once
class Ball {
  private:
    double x, y;
  public:
    Ball( const double x, const double y );
    Ball( const Ball& ball );
    Ball& operator= ( const Ball& ball );
    ~Ball();
};
```

属性（メンバ変数ともいう）

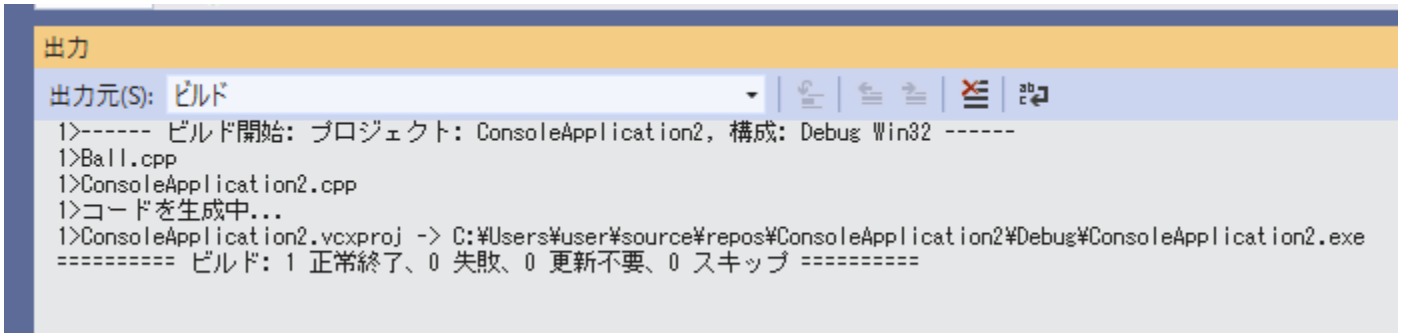
コンストラクタ，
デストラクタ

ソースコード

ファイル名: Ball.cpp

```
#include "Ball.h"
Ball::Ball( const double x, const double y ) : x( x ), y( y )
{
    /* do nothing */
}
Ball::Ball( const Ball& ball ) : x( ball.x ), y( ball.y )
{
    /* do nothing */
}
Ball& Ball::operator= (const Ball& ball )
{
    this->x = ball.x;
    this->y = ball.y;
    return *this;
}
Ball::~Ball()
{
    /* do nothing */
}
```

Visual Studio 2019 C++ でのビルド結果例



オブジェクトの生成

- **オブジェクト**を生成するプログラム

- クラス Ball のオブジェクト生成を行うプログラム

```
Ball* b = new Ball( 3, 4 );
```

メソッド

- **メソッド**は，**オブジェクトに属する操作や処理**のこと
- **引数（ひきすう）**とは，**メソッドに渡す値**のこと
- **メソッド**は，**クラス**に属する
- **属性**や**メソッド**にアクセスするときは**「.」や「->」を用いる**

例題２

- メソッド定義

例題２．メソッド

- Ball の座標値から，原点までの距離を求めるメソッド distance-to-0 を作り，実行する
- 演習１の Ball クラスを使用

ソースコード

ファイル名: Ball.h

```
#pragma once
class Ball {
  private:
    double x, y;
  public:
    Ball( const double x, const double y );
    Ball( const Ball& ball );
    Ball& operator= ( const Ball& ball );
    ~Ball();
    double distance_to_0() const;
};
```

属性（メンバ変数ともいう）

コンストラクタ，デストラクタ

追加されたメソッド

ソースコード

ファイル名: Ball.cpp

```
#include "Ball.h"
#include <math.h>
Ball::Ball( const double x, const double y ) : x( x ), y( y )
{
    /* do nothing */
}
Ball::Ball( const Ball& ball ) : x( ball.x ), y( ball.y )
{
    /* do nothing */
}
Ball& Ball::operator= (const Ball& ball )
{
    this->x = ball.x;
    this->y = ball.y;
    return *this;
}
Ball::~Ball()
{
    /* do nothing */
}
double Ball::distance_to_0() const
{
    return sqrt( ( this->x * this->x ) + ( this->y * this->y ) );
}
```

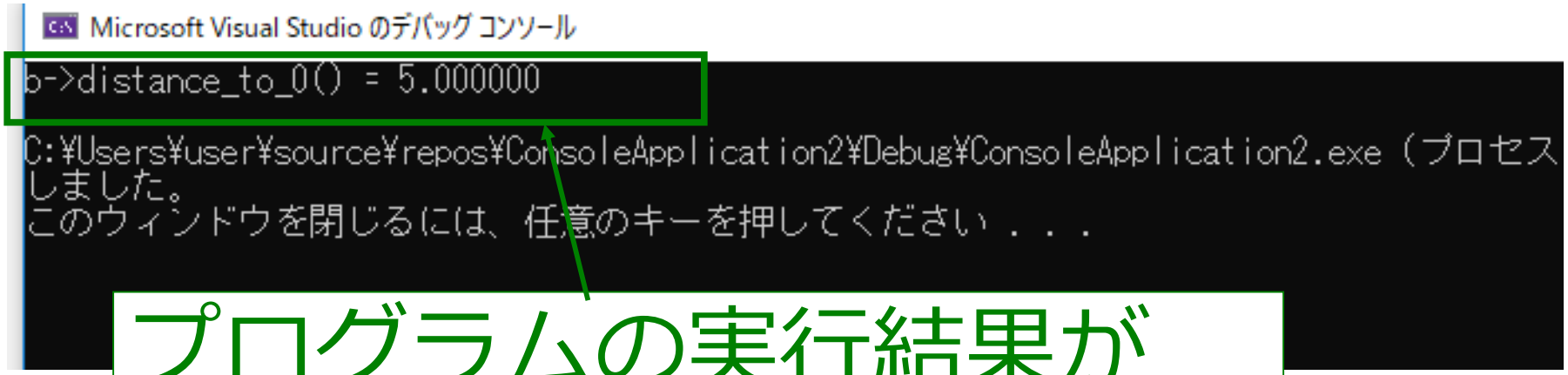
追加されたメソッド

ソースコード

ファイル名: main.cpp

```
#include <stdio.h>
#include "ball.h"
int main( int argc, char** argv )
{
  Ball* b = new Ball( 3, 4 );
  fprintf( stderr, "b->distance_to_0() = %f¥n", b->distance_to_0() );
  delete b;
}
```

Visual Studio 2019 C++ での実行結果例



プログラムの実行結果が
表示されている

まとめ

- **クラス定義**の中には，**属性**の定義（**属性名**と**データ型**），**コンストラクタ**の定義，**デストラクタ**の定義，その他**メソッド**の定義を含める
- **コンストラクタ**とは，**オブジェクト**の生成を行う**メソッド**のことである.
- **メソッド**は，**オブジェクトに属する操作や処理**のこと

- キーワード

class　　**クラス定義**

new　　**コンストラクタ**の呼び出し

実習

問題

- Student クラスを定義しなさい．メンバ変数は次の通り

```
int _age;
char _name[32];
```

解答例

ファイル名: Student.h

```
#pragma once
class Student {
private:
  int _age;
  char _name[32];
public:
  Student( const int age, const char name[32] );
  Student( const Student& student );
  Student& operator= (const Student& student );
  ~Student();
};
```

解答例

ファイル名: Student.cpp

```
#include <string.h>
#include "Student.h"
#pragma warning(disable:4996)
Student::Student( const int age, const char name[32] ) : _age( age )
{
  strcpy( this->_name, name );
}
Student::Student( const Student& student ) : _age( student._age )
{
  strcpy( this->_name, student._name );
}
Student& Student::operator= (const Student& student )
{
  this->_age = student._age;
  strcpy( this->_name, student._name );
  return *this;
}
Student::~Student()
{
  /* do nothing */
}
```